Threat Intel Report on

MITRE ATT&CK FRAMEWORK ENTERPRISE MATRICES

by

Team Cyber Ninjas

(Khushee Rane - 2049 , Sahil More - 2055 ,
Sameer More - 2056 , Surbhi Manani - 2072)

Table of Contents

1. Executive Summary

2. Framework Overview

3. Attack Chain Visualization

4. The 14 Enterprise Tactics
    - 4.1 Reconnaissance (TA0043)
    - 4.2 Resource Development (TA0042)
    - 4.3 Initial Access (TA0001)
    - 4.4 Execution (TA0002)
    - 4.5 Persistence (TA0003)
    - 4.6 Privilege Escalation (TA0004)
    - 4.7 Defense Evasion (TA0005)
    - 4.8 Credential Access (TA0006)
    - 4.9 Discovery (TA0007)
    - 4.10 Lateral Movement (TA0008)
    - 4.11 Collection (TA0009)
    - 4.12 Command and Control (TA0011)
    - 4.13 Exfiltration (TA0010)
    - 4.14 Impact (TA0040)

5. APT Group Case Studies

6. Detection and Response Framework

7. Mitigation Best Practices

8. Conclusion

9. References and Resources

Executive Summary

The MITRE ATT&CK® (Adversarial Tactics, Techniques, and Common Knowledge) framework is a globally accepted and highly authoritative knowledge base that systematically documents cyber adversary behavior observed in real-world attacks. Developed and maintained by the MITRE Corporation, the framework was publicly introduced in 2015 and has since evolved into the industry benchmark for analyzing, detecting, and mitigating cyber threats.

By focusing on adversary behavior rather than tools or malware alone, MITRE ATT&CK enables organizations to gain deeper visibility into how attackers operate across the entire attack lifecycle.

Key Framework Statistics (Version 18):

- **14 Tactics** defining adversary goals and objectives
- **216 Techniques** describing specific attack methods
- **475 Sub-techniques** offering detailed behavioral granularity
- **140+ Advanced Persistent Threat (APT) Groups** mapped with known TTPs
- **Three Core Matrices**:
    - Enterprise
    - Mobile
    - Industrial Control Systems (ICS)

These statistics highlight the depth, maturity, and real-world relevance of the ATT&CK framework.

Purpose of This Guide:

This document serves as a practical and operational reference designed for a wide range of cybersecurity professionals, including:

- **Blue Teams** aiming to strengthen detection and monitoring capabilities
- **Red Teams** developing realistic adversary simulations and attack scenarios
- **Security Architects** evaluating defensive architectures and control coverage
- **Incident Responders** conducting threat investigations and forensic analysis
- **Risk and Governance Professionals** assessing organizational exposure and resilience

Document Scope:

This guide focuses exclusively on the **Enterprise ATT&CK Matrix**, covering all **14 enterprise tactics**. For each tactic, the guide provides:

- A clear explanation of the tactic's objective
- Two representative techniques with official MITRE IDs
- Real-world examples linked to known APT groups
- Detection approaches and telemetry sources
- Recommended mitigation strategies
- Visual attack-chain representations to enhance understanding

Framework Overview

What is MITRE ATT&CK?

MITRE ATT&CK is an open-source, community-driven knowledge base that catalogs adversary tactics and techniques derived from the analysis of real-world cyber intrusions. It offers a structured and consistent approach to understanding how attackers plan, execute, and maintain cyber operations against information systems.

Rather than focusing on vulnerabilities alone, ATT&CK emphasizes **attacker behavior**, making it highly effective for threat modeling, detection engineering, and defensive strategy development.

Core Components of the Framework:

1. Tactics – *The "Why"*

- Represent the adversary's high-level objectives
- Define **14 distinct tactical goals** in the Enterprise Matrix
- Reflect a logical progression through the cyber attack lifecycle

2. Techniques – *The "How"*

- Describe the specific methods adversaries use to achieve tactical goals
- **216 techniques** are documented within the Enterprise Matrix
- Each technique is assigned a unique identifier (e.g., **T1059**)

3. Sub-techniques – *The "Details"*

- Provide granular variations of parent techniques
- **475 sub-techniques** enable precise behavioral classification
- Identified using the format **T####.###** (e.g., **T1566.001**)

4. Procedures – *The "Reality"*

- Real-world implementations of techniques by threat actors
- Mapped to specific campaigns and APT groups
- Supported by evidence-based threat intelligence reporting

Framework Applications:

Use Case	Description	Benefit
Threat Detection	Map alerts and logs to ATT&CK techniques	Improve SIEM and EDR coverage
Red Teaming	Simulate realistic adversary behavior	Validate defenses against real TTPs
Gap Analysis	Evaluate control and detection coverage	Identify defensive blind spots
Threat Intelligence	Standardize threat actor profiling	Enable effective information sharing
Incident Response	Classify adversary behavior during incidents	Accelerate investigation timelines
Security Validation	Test detection and response capabilities	Measure security effectiveness

Complementary Frameworks:

MITRE D3FEND acts as a defensive counterpart to ATT&CK by focusing on:

- Defensive techniques and countermeasures
- Detection and prevention strategies mapped to ATT&CK techniques
- Answering *“How to stop the attacker”* versus ATT&CK’s focus on *“What the attacker does”*

Attack Chain Visualization

Complete MITRE ATT&CK Enterprise Attack Chain Flow

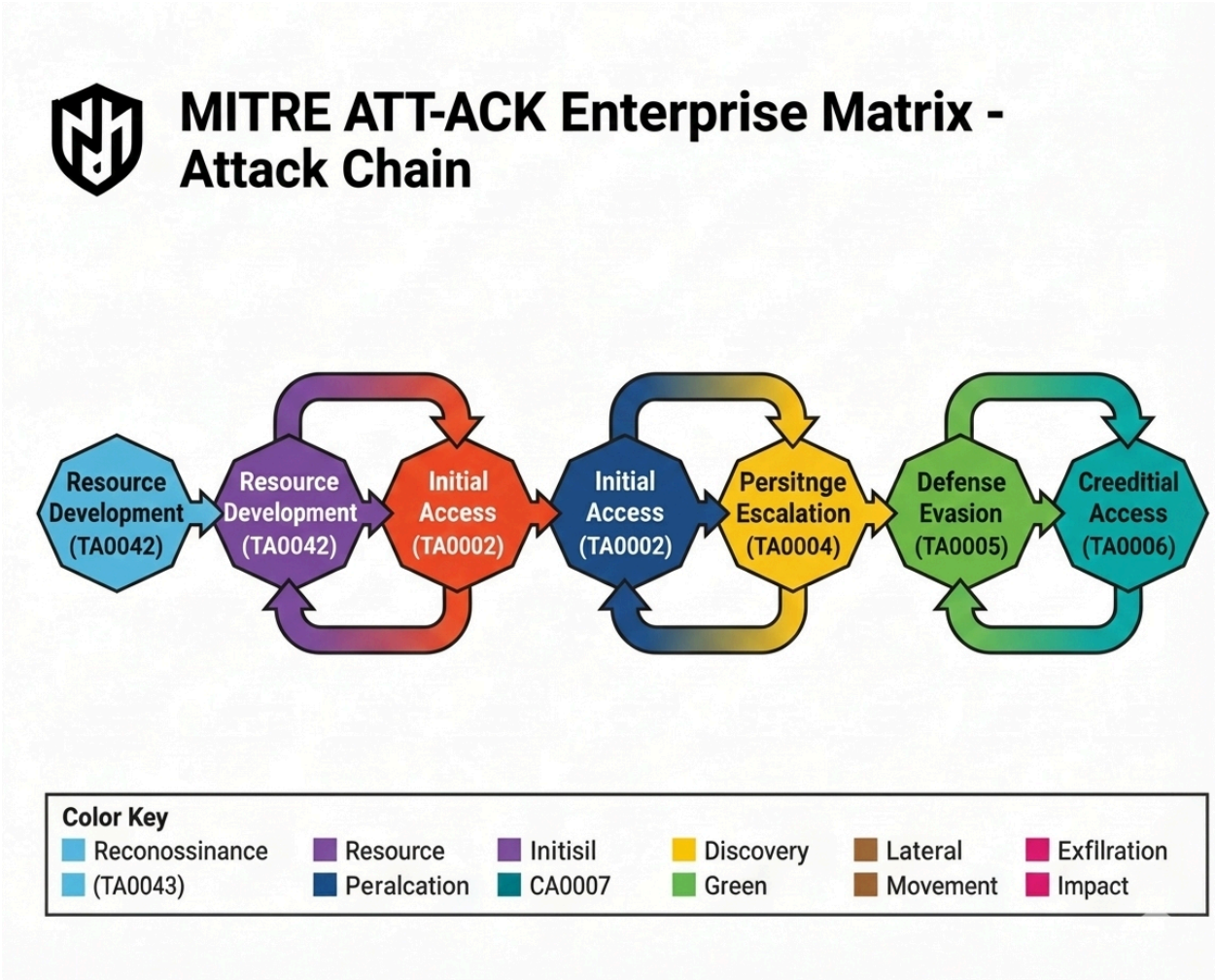

Figure 1: The 14 tactics representing typical adversary progression from reconnaissance to impact

Tactic Relationships and Dependencies

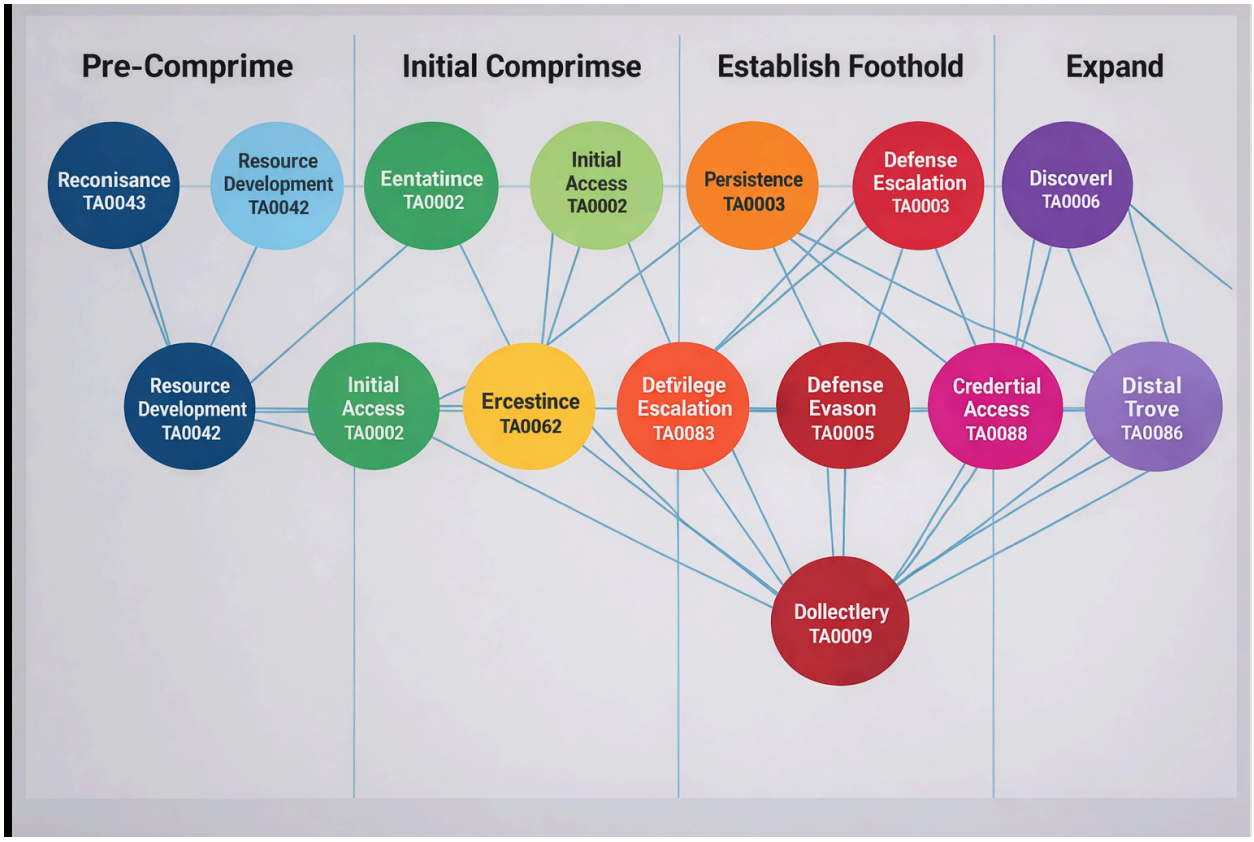

Figure 2: Network graph showing how tactics interconnect during sophisticated attacks

Attack Lifecycle Phases

Figure 3: Grouping of tactics into pre-compromise, compromise, and postcompromise phases

Attacker vs Defender Perspective

Figure 4: Dual perspective showing offensive actions and defensive responses

The 14 Enterprise Tactics

1. Reconnaissance (TA0043)

Tactic Objective

The adversary seeks to collect information that can support planning and preparation for future malicious activities.

Tactic Description

Reconnaissance includes methods used by adversaries to **collect intelligence about a target environment before compromise**. These activities may be conducted **actively**, by directly interacting with systems, or **passively**, by collecting publicly available information.
Information gathered during this stage can include organizational structure, employee details, network infrastructure, security controls, and exposed services. The intelligence obtained enables attackers to **select attack vectors, tailor exploits, prioritize targets**, and guide subsequent phases such as Initial Access and Credential Access.

Tactic ID

TA0043

Total Techniques

11

Typical Phase

Pre-compromise

ATT&CK Version

Created October 2020

Technique 1: Active Scanning (T1595)

Description

Active Scanning involves directly probing a target's infrastructure to identify live systems, exposed services, misconfigurations, and vulnerabilities. Unlike passive techniques, active scanning generates network traffic and interacts with victim systems, increasing the chance of detection.

Sub-techniques

- **T1595.001 – Scanning IP Blocks:** Identifying active hosts across IP ranges
- **T1595.002 – Vulnerability Scanning:** Using automated tools to discover known weaknesses
- **T1595.003 – Wordlist Scanning:** Enumerating directories, files, subdomains, or virtual hosts using wordlists

Real-World Example

Volt Typhoon (China, 2024)
Volt Typhoon conducted extensive active scanning against U.S. critical infrastructure organizations, including energy and water utilities. The group scanned public-facing routers, VPN gateways, and network appliances to identify exposed services and outdated firmware. These scans helped them locate misconfigured systems that later facilitated long-term persistence within operational technology (OT) environments.

Detection Methods

- Monitor abnormal inbound network traffic patterns
- Detect port scanning activity via firewall and IDS alerts

- Identify scanning tools such as Nmap, Masscan, or custom scripts
- Analyze DNS logs for excessive subdomain resolution attempts
- Review web server logs for directory or endpoint brute-force behavior

Mitigation Strategies

Mitigation	Description	Priority
Network Segmentation	Reduce exposure of sensitive systems to public networks	High
IDS/IPS Deployment	Detect and alert on scanning behavior	High
Rate Limiting	Throttle repeated requests to public services	Medium
Honeypots	Use decoy systems to detect and study scanners	Medium
External Threat Monitoring	Monitor scanning activity from known malicious IP ranges	High

Technique 2: Gather Victim Identity Information (T1589)

Description

Adversaries collect identity-related information about individuals associated with the target organization. This data supports targeting decisions and enables social engineering, credential theft, and phishing campaigns.

Sub-techniques

- **T1589.001 – Credentials:** Obtaining leaked or reused login information

- **T1589.002 – Email Addresses:** Collecting corporate or personal email IDs

- **T1589.003 – Employee Names:** Identifying staff names, job roles, and reporting structures

Real-World Example

FIN7 (Carbanak Group)
FIN7 conducted large-scale identity reconnaissance against retail and hospitality companies by scraping corporate websites, professional networking platforms, and breached data repositories. The group built detailed employee profiles, including finance and IT personnel, which were later used to launch highly convincing phishing campaigns that resulted in malware deployment and payment system compromise.

Detection Methods

- Monitor unusual scraping or crawling of employee directories

- Track excessive OSINT-related queries referencing the organization

- Detect abnormal access patterns to public-facing staff information

- Identify automated data harvesting on corporate websites

- Monitor third-party breach intelligence sources for leaked identity data

Mitigation Strategies

Mitigation	Description	Priority
Information Exposure Reduction	Minimize public employee details online	High
Dark Web Monitoring	Detect leaked credentials early	High
Credential Rotation	Reset exposed or suspected credentials	High
Employee Awareness Training	Promote safe social media practices	Medium
Data Loss Prevention (DLP)	Prevent leakage of internal documents	Medium

2. Resource Development (TA0042)

Tactic Objective

The adversary seeks to establish and maintain resources that can be used to support future malicious operations.

Tactic Description

Resource Development includes techniques used by adversaries to create, acquire, compromise, or steal resources that enable targeting and execution of attacks. These resources may include infrastructure (servers, domains, cloud services), accounts, tools, malware, or technical capabilities.
Unlike Reconnaissance, which focuses on gathering information, Resource Development actively prepares the operational foundation required for successful intrusion, persistence, and command-and-control activities. Actions taken during this phase typically occur before direct compromise of the victim environment.

Tactic ID

TA0042

Total Techniques

8

Typical Phase

Pre-compromise

ATT&CK Version

Created October 2020

Technique 1: Acquire Infrastructure (T1583)

Description

Adversaries acquire infrastructure that can be used to host malware, conduct phishing campaigns, operate command-and-control (C2) servers, or anonymize malicious traffic. Infrastructure may be purchased, leased, compromised, or covertly obtained using third-party services. This enables attackers to operate at scale while obscuring attribution.

Sub-techniques

- **T1583.001 – Domains:** Registering domains for phishing, malware delivery, or C2
- **T1583.002 – DNS Server:** Establishing malicious DNS infrastructure
- **T1583.003 – Virtual Private Server:** Renting VPS instances for attack operations
- **T1583.004 – Server:** Using physical or cloud servers
- **T1583.005 – Botnet:** Renting or purchasing botnet services
- **T1583.006 – Web Services:** Abusing legitimate web hosting platforms
- **T1583.008 – Malvertising:** Using malicious online advertisements

Real-World Example

Scattered Spider (UNC3944, 2023–2024)
Scattered Spider acquired cloud-based VPS infrastructure and registered look-alike domains impersonating technology and telecom companies. These domains were used for phishing and credential harvesting

campaigns targeting IT help desks. The group leveraged legitimate cloud providers to host infrastructure, enabling rapid scaling and evasion of traditional IP-based blocking.

Detection Methods

- Monitor newly registered domains resembling organizational assets
- Detect typosquatting and homograph attacks
- Analyze DNS traffic for fast-flux or domain generation algorithms (DGA)
- Identify outbound connections to suspicious VPS providers
- Monitor SSL certificate transparency logs for impersonation domains

Mitigation Strategies

Mitigation	Description	Priority
Domain Monitoring	Detect newly registered look-alike domains	High
Threat Intelligence	Track known malicious infrastructure indicators	High
Certificate Transparency Monitoring	Identify fraudulent SSL certificates	High
Domain Takedown	Establish rapid takedown procedures	Medium
Brand Protection	Register defensive domains	Medium

Technique 2: Develop Capabilities (T1587)

Description

Adversaries may internally develop malicious tools and capabilities rather than purchasing them. This includes writing custom malware, building exploits, creating encryption tools, and developing command-and-control frameworks. Internally developed capabilities allow attackers greater control, flexibility, and evasion from signature-based defenses.

Sub-techniques

- **T1587.001 – Malware:** Developing custom malicious software
- **T1587.002 – Code Signing Certificates:** Creating or stealing certificates
- **T1587.003 – Digital Certificates:** Generating SSL/TLS certificates
- **T1587.004 – Exploits:** Developing zero-day or known exploits

Real-World Example

MuddyWater (Iran, 2022–2024)
MuddyWater developed custom PowerShell-based malware frameworks designed for reconnaissance, lateral movement, and data exfiltration. The group frequently modified payloads between campaigns, using encrypted C2 channels and obfuscated scripts to evade detection. Their in-house tooling allowed rapid adaptation against updated security controls in Middle Eastern government and telecom sectors.

Detection Methods

- Identify previously unseen malware families
- Monitor abuse of code-signing certificates
- Detect repeated recompilation or polymorphic malware variants
- Track exploit testing behavior in sandbox environments

- Analyze similarities between custom malware samples

Mitigation Strategies

Mitigation	Description	Priority
Threat Intelligence Sharing	Participate in industry information sharing	High
Malware Analysis	Maintain sandbox environments	High
Code Signing Validation	Verify all signed executables	High
Exploit Mitigation	Deploy exploit prevention technologies	High
Certificate Pinning	Protect critical applications	Medium

3. Initial Access (TA0001)

Tactic Objective

The adversary seeks to gain an initial foothold within the target network.

Tactic Description

Initial Access techniques represent the methods used by adversaries to enter a target environment. These techniques exploit human trust, software vulnerabilities, or exposed services to gain access. Successful initial access enables attackers to deploy payloads, establish persistence, and move laterally within the network.

Tactic ID

TA0001

Total Techniques

11

Typical Phase

Initial compromise

ATT&CK Version

Created October 2018

Technique 1: Phishing (T1566)

Description

Phishing is a widely used social engineering technique in which attackers send deceptive messages to victims with the goal of stealing sensitive information, gaining unauthorized access, or delivering malware. These messages are typically delivered through email but may also use messaging platforms, cloud services, voice calls, or collaboration tools.

Attackers often impersonate **trusted entities** such as banks, IT support teams, government agencies, or well-known service providers. Messages are carefully crafted to create urgency, fear, or curiosity, prompting victims to click malicious links, open infected attachments, or share login credentials. In advanced phishing campaigns, messages are **highly targeted (spearphishing)** and customized based on the victim's role, organization, or recent activities.

Phishing is frequently used as an **initial access vector** and can lead to credential theft, malware execution, lateral movement, data exfiltration, and long-term persistence within a network.

Sub-techniques

- **T1566.001 – Spearphishing Attachment**
- **T1566.002 – Spearphishing Link**
- **T1566.003 – Spearphishing via Service**
- **T1566.004 – Spearphishing Voice**

Real-World Example

1. Storm-0558 (2023)

Storm-0558, a China-linked threat actor, conducted highly targeted phishing campaigns against multiple government agencies. The attackers used **forged authentication tokens** to gain access to Outlook Web Access (OWA) accounts. Victims were lured into clicking malicious links embedded in phishing emails, which allowed the attackers to bypass **Multi-Factor Authentication (MFA)** protections. This resulted in **long-term email surveillance**, access to sensitive communications, and data exfiltration without triggering immediate detection.

2. Google & Facebook Business Email Compromise (2013–2015)

Attackers impersonated a legitimate hardware vendor and sent phishing emails containing fraudulent invoices to employees at Google and Facebook. These emails appeared authentic and were part of a **spearphishing attachment and link-based campaign**. Over time, employees unknowingly transferred payments to attacker-controlled bank accounts, resulting in losses exceeding **$100 million**. This case highlights how phishing can be used for **financial fraud** rather than malware delivery.

3. Twitter Bitcoin Scam (2020)

Attackers used phishing techniques to target Twitter employees by impersonating internal IT support. Through **spearphishing links**, attackers captured credentials and accessed internal administrative tools. This allowed them to compromise high-profile accounts, including those of Elon Musk, Barack Obama, and Apple, to promote a cryptocurrency scam. The incident demonstrated how phishing can lead to **privileged access abuse** and large-scale public impact.

Detection Methods

- **Analyze email headers for authentication failures**
  Check SPF, DKIM, and DMARC results to identify spoofed sender domains or unauthorized mail servers.

- **Detect suspicious attachments and embedded macros**
  Scan attachments for known malware signatures, unusual file types, or malicious macro behavior, especially in Office documents.

- **Monitor user click behavior on URLs**
  Track clicks on shortened URLs, newly registered domains, or domains with poor reputation scores.

- **Identify anomalous outbound connections after email interaction**
  Look for unexpected network connections to command-and-control (C2) servers or unfamiliar external IP addresses following email engagement.

- **Enable user-reported phishing workflows**
  Encourage employees to report suspicious emails using built-in reporting tools, allowing faster incident response and improved security awareness.

Mitigation Strategies

Mitigation	Description	Priority
Email Authentication	Enforce DMARC, SPF, DKIM	Critical
Security Awareness Training	Conduct phishing simulations	Critical
Advanced Email Filtering	Block malicious emails	Critical
Attachment Sandboxing	Detonate suspicious files	High
Multi-Factor Authentication	Protect email access	Critical

Technique 2: Exploit Public-Facing Application (T1190)

Description

Adversaries exploit vulnerabilities in **public-facing applications** to gain **initial access** to a target environment. Public-facing applications are systems accessible directly from the internet, such as **web servers, APIs, VPN gateways, email servers, and content management systems (CMS)**.

Attackers commonly take advantage of:

- **Unpatched software vulnerabilities**
- **Outdated CMS plugins or themes**
- **Weak input validation**
- **Misconfigured authentication mechanisms**

- **Known Common Vulnerabilities and Exposures (CVEs)**

Once a vulnerability is successfully exploited, attackers may achieve **remote code execution (RCE)**, **web shell deployment**, or **unauthorized access**, allowing them to move further into the network. This technique is frequently used during the **initial access phase** of an attack lifecycle.

Commonly Targeted Applications

- Web servers (Apache, Nginx, IIS)
- CMS platforms (WordPress, Joomla, Drupal)
- Web frameworks and APIs
- VPN appliances
- Enterprise portals and dashboards

Real-World Example

1. LockBit Affiliates (2024)

LockBit ransomware affiliates exploited **unpatched web applications** running **outdated CMS plugins** exposed to the internet. By abusing known vulnerabilities, attackers gained **initial access to enterprise environments**.
After access was established, the compromised servers were used to:

- Deploy ransomware payloads
- Establish persistence
- Perform **lateral movement** within internal networks
  This attack demonstrated how neglected patch management can

lead directly to full network compromise.

2. MOVEit Transfer Exploitation (Cl0p Ransomware Group – 2023–2024)

The Cl0p ransomware group exploited a **zero-day SQL injection vulnerability** in the **MOVEit Transfer** file transfer application, which was publicly accessible.
Attackers:

- Extracted sensitive data from affected organizations
- Deployed web shells for continued access
- Used the breach primarily for **data exfiltration and extortion**, rather than immediate ransomware deployment
  This incident impacted **hundreds of organizations worldwide**, highlighting the severe risk posed by vulnerabilities in internet-facing applications.

Detection Methods

1. Monitor Web Logs for Exploit Indicators

- Analyze HTTP and HTTPS logs for suspicious patterns such as:
  - SQL injection attempts
  - Command injection strings
  - Directory traversal attempts
- Look for abnormal request parameters and encoded payloads.

2. Deploy Web Application Firewalls (WAF)

- Use WAFs to detect and block:
    - Known attack signatures
    - Malicious payloads targeting application vulnerabilities
- Enable real-time alerts for repeated exploitation attempts.

3. Detect Abnormal HTTP Requests

- Identify unusual request methods (e.g., unexpected `POST`, `PUT`, or `DELETE`)
- Monitor spikes in traffic from single IP addresses
- Flag unexpected file uploads or script execution attempts.

4. Track Exploitation of Known CVEs

- Continuously scan public-facing assets for known vulnerabilities
- Correlate detected CVEs with:
    - Threat intelligence feeds
    - Active exploitation campaigns
- Prioritize patching of **high-severity and actively exploited CVEs**.

Mitigation Strategies

Mitigation	Description	Priority
Patch Management	Apply updates promptly	Critical
Web Application Firewall	Block exploit attempts	Critical
Network Segmentation	Isolate web servers	High
Input Validation	Sanitize user input	Critical

4. Execution (TA0002)

Tactic Objective

The adversary seeks to execute malicious code.

Tactic Description

Execution techniques allow adversaries to run attacker-controlled code on compromised systems. This tactic bridges initial access with post-compromise objectives such as persistence, credential theft, and lateral movement.

Tactic ID

TA0002

Total Techniques

17

Typical Phase

Post-compromise

ATT&CK Version

Created October 2018

Technique 1: Command and Scripting Interpreter (T1059)

Description

Adversaries abuse built-in command-line interfaces and scripting interpreters provided by the operating system to execute malicious commands. These interpreters are **trusted system components**, which allows attackers to blend in with normal administrative activity and evade security controls.

Sub-techniques

- **T1059.001 – PowerShell**
- **T1059.003 – Windows Command Shell**
- **T1059.004 – Unix Shell**
- **T1059.006 – Python**

Real-World Example

FIN12 (2023)

FIN12 used **PowerShell scripts** to rapidly deploy ransomware across compromised enterprise environments. Their campaigns relied heavily on **encoded PowerShell commands** to disable endpoint security tools, move laterally, and execute ransomware payloads. The group focused on speed, often completing attacks within hours of initial access.

APT29 / Cozy Bear

APT29 has repeatedly leveraged **PowerShell and Python scripts** to perform stealthy reconnaissance and persistence in targeted networks. Their attacks used in-memory PowerShell execution to avoid detection and

Python-based tooling for data exfiltration and command-and-control operations.

Detection Methods

Enable PowerShell Logging

- Script Block Logging
- Module Logging
- Transcription Logging

Detect Encoded or Obfuscated Commands

- Monitor for Base64-encoded PowerShell commands
- Identify excessive use of `-EncodedCommand`, `Invoke-Expression`, or `IEX`

Monitor Suspicious Process Trees

- Parent-child relationships such as `winword.exe → powershell.exe`
- Unexpected shell execution from non-administrative applications

Track Script Execution from Temporary Locations

- Execution of scripts from `%TEMP%`, `/tmp`, or user-writable directories
- Unusual script execution by standard user accounts

Mitigation Strategies

Mitigation	Description	Priority
PowerShell Logging	Enable full logging	Critical
Constrained Language Mode	Restrict script capabilities	High
Script Signing	Require signed scripts	High
Application Control	Restrict execution	High

Technique 2: Process Injection (T1055)

Description

Process Injection is a post-exploitation technique where attackers execute malicious code within the address space of a legitimate, running process. By injecting code into trusted system or application processes (such as browsers, system services, or security software), attackers can evade endpoint security controls, blend in with normal activity, escalate privileges, and maintain persistence.

This technique is widely used in advanced malware, credential stealers, ransomware, and APT campaigns because it allows malicious activity to run under the context of a trusted process, making detection significantly more difficult.

Sub-techniques

- **T1055.001 – Dynamic-link Library Injection**
  Injecting malicious DLLs into legitimate running processes to execute attacker-controlled code while evading detection.

- **T1055.002 – Portable Executable Injection**
  Injecting malicious Portable Executable (PE) files directly into the memory of a target process without writing them to disk.

- **T1055.003 – Thread Execution Hijacking**
  Manipulating or hijacking existing threads within a process to redirect execution flow to malicious code.

- **T1055.004 – Asynchronous Procedure Call (APC)**
  Executing malicious code by queuing APCs to existing threads, allowing stealthy execution within trusted processes.

- **T1055.012 – Process Hollowing**
  Launching a legitimate process in a suspended state and replacing its memory contents with malicious payloads before resuming execution.

- **T1055.013 – Process Doppelgänging**
  Using NTFS transactions to create and execute malicious processes that appear legitimate, bypassing traditional file-based detection mechanisms.

Real-World Example

RedLine Stealer (2024)

RedLine injected malicious code into browser processes to harvest credentials and session tokens while bypassing antivirus detection. The injected code monitored browser memory to extract usernames, passwords, cookies, and autofill data without triggering alerts.

Emotet Malware

Emotet used DLL injection and process hollowing to inject its payload into trusted Windows processes such as `explorer.exe`. This allowed Emotet to spread laterally, download additional malware, and evade endpoint detection systems.

Cobalt Strike Beacons

Cobalt Strike frequently uses APC injection, PE injection, and thread hijacking to run command-and-control beacons inside legitimate processes. This makes detection difficult, as network traffic and execution appear to originate from trusted applications.

TrickBot

TrickBot employed process hollowing and DLL injection to hide its banking trojan modules within system processes, enabling credential theft and persistence while avoiding behavioral detection.

Detection Methods

Monitor Cross-Process Memory Access

- Detect suspicious use of APIs such as `WriteProcessMemory`, `ReadProcessMemory`, and `CreateRemoteThread`
- Flag unauthorized access between unrelated processes

Detect Abnormal DLL Injections

- Monitor unusual DLL loads in sensitive processes

- Identify DLLs loaded from non-standard directories or temporary locations

Analyze Suspicious Thread Creation

- Identify threads created in remote processes
- Detect thread start addresses pointing to non-module memory regions

Additional Advanced Detection Techniques

- Behavioral analysis using EDR solutions
- Memory scanning for injected code regions
- Monitoring process parent-child anomalies
- Correlating process injection with network callbacks

Mitigation Strategies

Mitigation	Description	Priority
Endpoint Detection & Response	Detect injection behavior	Critical
Memory Protection	Enable DEP and ASLR	High
API Monitoring	Detect suspicious calls	High

5. Persistence (TA0003)

Tactic Objective

The adversary seeks to maintain long-term access.

Tactic Description

Persistence techniques allow adversaries to retain access across reboots, credential changes, and system updates.

Tactic ID

TA0003

Total Techniques

23

Typical Phase

Post-compromise

ATT&CK Version

Created October 2018

Technique 1: Boot or Logon Autostart Execution (T1547)

Description

Boot or Logon Autostart Execution is a persistence technique used by adversaries to ensure that malicious code is automatically executed whenever a system boots or a user logs on. By embedding malware into legitimate startup mechanisms of the operating system, attackers can maintain long-term access to compromised systems even after reboots or user logouts. This technique is commonly used in post-exploitation phases to maintain persistence and evade detection.

Sub-techniques:

- **T1547.001 – Registry Run Keys / Startup Folder**
  Adding entries to Windows registry or startup folder

- **T1547.002 – Authentication Package**
  Modifying authentication DLLs

- **T1547.004 – Winlogon Helper DLL**
  Abusing Winlogon registry keys

- **T1547.005 – Security Support Provider**
  Injecting SSP DLLs

- **T1547.009 – Shortcut Modification**
  Modifying LNK files

- **T1547.012 – Print Processors**
  Installing malicious print processors

- **T1547.013 – XDG Autostart Entries**
  Linux autostart abuse

Real-World Example

OilRig (APT34, 2022)

OilRig modified registry run keys and startup scripts to maintain persistent access to compromised government networks in the Middle East. This allowed the group to re-establish control after system reboots and continue espionage operations over extended periods.

APT29 (Cozy Bear)

APT29 leveraged registry Run keys and Winlogon helper DLLs to maintain persistence on Windows systems during espionage campaigns targeting diplomatic and government organizations. The use of trusted system components helped evade detection.

FIN7

The FIN7 cybercrime group abused shortcut modification and registry autostart locations to ensure point-of-sale malware was executed whenever systems were restarted, enabling long-term financial data theft.

Detection Methods

- **Monitor registry autostart locations** such as Run, RunOnce, Winlogon, and Services keys for unauthorized or suspicious entries.
- **Track startup folder changes** and detect unexpected executable files or scripts added to user or system startup directories.
- **Audit scheduled tasks** and correlate them with startup-related execution behavior.
- **Inspect DLL loading behavior** for authentication packages, SSPs, and print processors.

- **Use endpoint detection and response (EDR) tools** to detect anomalous processes executed during boot or logon events.

Mitigation Strategies

Mitigation	Description	Priority
Registry Monitoring	Alert on autostart changes	Critical
Least Privilege	Limit admin access	Critical
Regular Audits	Review startup entries	Medium

Technique 2: Create or Modify System Process (T1543)

Description

Create or Modify System Process (T1543) is a persistence and privilege escalation technique used by attackers to execute malicious payloads by creating new system services or modifying existing ones. These system processes typically run with **elevated (administrator/root) privileges**, allowing attackers to maintain long-term access, survive system reboots, and execute malware stealthily.

Attackers abuse legitimate service management mechanisms across different operating systems (Windows, Linux, macOS) to disguise malicious activity as trusted system components.

Sub-techniques:

- **T1543.001 – Launch Agent**
  macOS launch agents for persistence

- **T1543.002 – Systemd Service**
  Linux systemd service creation

- **T1543.003 – Windows Service**
  Creating malicious Windows services

- **T1543.004 – Launch Daemon**
  macOS launch daemons

Real-World Example

Conti Ransomware (2021–2022)

Conti ransomware operators created **malicious Windows services** disguised as system update services to maintain persistence after initial infection. These services ensured the ransomware payload continued running even after reboots and allowed attackers to re-enter compromised systems during lateral movement and data exfiltration phases.

Ryuk Ransomware

Ryuk used malicious Windows services to deploy ransomware payloads and maintain control over infected systems. The services were often named to resemble legitimate Windows components, making detection difficult for administrators.

SolarWinds Supply Chain Attack (2020)

Attackers modified legitimate system services to load malicious DLLs during service execution. This allowed the malware to persist and operate

with elevated privileges while appearing as part of trusted system processes.

APT29 (Cozy Bear)

APT29 leveraged system services and scheduled system processes to maintain long-term persistence in targeted enterprise environments, especially on Windows-based infrastructure.

Detection Methods

Monitor Service Creation Events

- Track new or modified services using system logs and security tools
- Alert on unexpected service installations
- Monitor use of service creation utilities (`sc.exe`, `systemctl`, `launchctl`)

Analyze Unusual Service Paths

- Identify services running from non-standard directories such as:
    - `Temp`
    - `AppData`
    - User home directories
- Legitimate services usually run from system directories (`System32`, `/usr/bin`, `/Library`)

Additional Detection Techniques

- Baseline known legitimate services and compare changes
- Monitor startup persistence mechanisms
- Use Endpoint Detection and Response (EDR) solutions
- Correlate service creation with suspicious network activity

Mitigation Strategies

Mitigation	Description	Priority
Service Auditing	Review installed services	High
Code Signing Validation	Allow only signed services	High
Behavioral Monitoring	Detect service abuse	Critical

6. Privilege Escalation (TA0004)

Tactic Objective

The adversary seeks to gain elevated permissions.

Tactic Description

Privilege Escalation enables attackers to gain higher-level access, allowing them to control systems, disable security tools, and access sensitive data.

Tactic ID

TA0004

Total Techniques

14

Typical Phase

Post-compromise

ATT&CK Version

Created October 2018

Technique 1: Exploitation for Privilege Escalation (T1068)

Description

Exploitation for Privilege Escalation refers to the techniques used by attackers to exploit software vulnerabilities in order to elevate their access level on a compromised system. Typically, attackers begin with limited privileges (such as a standard user account) and leverage flaws in the operating system, kernel, drivers, or privileged applications to gain **administrator or SYSTEM-level access**.

Once elevated privileges are obtained, attackers can:

- Disable or bypass security controls
- Install persistent malware
- Access sensitive system resources
- Move laterally within the network
- Execute destructive actions such as ransomware deployment

This technique is especially dangerous because it allows attackers to turn an initial foothold into **full system control**.

Common Vulnerabilities Exploited:

- Kernel exploits (Windows, Linux)
- UAC bypass vulnerabilities
- Token impersonation vulnerabilities

- DLL hijacking in privileged processes
- Unquoted service path vulnerabilities

Real-World Example

BlackCat (ALPHV, 2024)

BlackCat ransomware affiliates exploited Windows privilege escalation vulnerabilities after gaining initial access. By elevating privileges to SYSTEM level, they disabled endpoint security solutions and deployed ransomware across enterprise networks, causing large-scale operational disruption.

LockBit Ransomware (2023–2024)

LockBit operators leveraged local privilege escalation vulnerabilities in Windows environments to gain administrative access. This allowed them to terminate security processes, establish persistence, and encrypt systems at scale within corporate networks.

Detection Methods

Monitor Elevated Process Creation

- Track unexpected processes running with administrator or SYSTEM privileges, especially when launched by non-privileged parent processes.

Detect Kernel Exploit Attempts

- Use endpoint detection and response (EDR) tools to identify abnormal kernel behavior, suspicious driver loading, or exploitation of known kernel vulnerabilities.

Track Abnormal Token Manipulation

- Monitor for unusual access token changes, token impersonation events, or processes requesting higher privileges without legitimate justification.

Mitigation Strategies

Mitigation	Description	Priority
Patch Management	Apply updates rapidly	Critical
Exploit Protection	Enable OS protections	Critical
Least Privilege	Restrict user permissions	Critical

Technique 2: Valid Accounts (T1078)

Description

Adversaries abuse **legitimate but compromised credentials** to gain initial access, escalate privileges, maintain persistence, and move laterally within a target environment. Because valid accounts are commonly used by legitimate users, malicious activity can blend in with normal operations, making detection difficult. Attackers may obtain credentials through phishing, credential dumping, malware, password reuse, or third-party breaches.

Once access is achieved, adversaries can perform actions such as accessing sensitive data, modifying configurations, disabling security controls, or impersonating trusted users and administrators.

Sub-techniques:

- **T1078.001 – Default Accounts:** Exploiting default admin/root credentials
- **T1078.002 – Domain Accounts:** Abusing Active Directory credentials
- **T1078.003 – Local Accounts:** Exploiting local administrator accounts
- **T1078.004 – Cloud Accounts:** Compromising privileged cloud accounts (AWS, Azure, GCP)

Real-World Example

Okta Support Breach (2023)

Attackers abused **stolen Okta support credentials** to gain unauthorized access to administrative consoles. This enabled privilege escalation and access to customer environments, affecting multiple organizations that relied on Okta for identity management.

Uber Breach (2022)

Attackers used **stolen credentials combined with MFA fatigue attacks** to gain access to an Uber employee's account. Once inside, they accessed internal systems, administrative tools, and sensitive data, demonstrating how valid account abuse can bypass strong authentication mechanisms.

Detection Methods

Monitor Abnormal Login Behavior

- Identify logins from unusual geolocations or IP addresses
- Detect access outside normal working hours
- Monitor impossible travel scenarios

Detect MFA Fatigue Attempts

- Alert on repeated MFA push notifications
- Monitor excessive authentication failures followed by success
- Implement number-matching or phishing-resistant MFA

Track Privileged Account Usage

- Monitor use of administrator and service accounts
- Alert on privilege escalation events
- Detect access to sensitive systems or configurations

Additional Defensive Measures

- Enforce least privilege access
- Regularly rotate credentials and API keys
- Disable or rename default accounts

- Implement conditional access policies

Mitigation Strategies

Mitigation	Description	Priority
Multi-Factor Authentication	Protect privileged accounts	Critical
Privileged Access Workstations	Isolate admin activity	High
Just-in-Time Access	Limit privilege duration	High

7. Defense Evasion (TA0005)

Tactic Objective:

The adversary is trying to avoid detection by security controls, such as Web Application Firewalls (WAFs), API Gateways, and Logging/SIEM systems, while interacting with the API.

Tactic Description:

In the context of APIs, "Defense Evasion" is primarily characterized by Request Manipulation and Architectural Evasion. Unlike traditional evasion which might involve clearing system event logs or masquerading processes on an endpoint, API evasion focuses on modifying the traffic payload to render it invisible or benign to inspection intermediaries while remaining executable by the backend logic. Adversaries exploit "Parsing Discrepancies" between the security layer (WAF) and the application layer (API Parser). If a WAF normalizes a request differently than the backend server, malicious payloads can slip through the gap. Furthermore, adversaries leverage "Shadow" or "Zombie" APIs— endpoints that are active but unmonitored—to conduct operations outside the visibility of standard governance protocols. The goal is to render malicious traffic indistinguishable from legitimate API calls or to direct traffic through channels where inspection logic is absent or misconfigured.

Tactic ID:

TA0005

Total Techniques:

47

Typical Phase:
Intrusion / Exploitation ATT&CK

Version:
v18 (Adapted for API)

Technique 1: Obfuscated Files or Information (T1027) - Adapted to "WAF Bypass via Payload Manipulation"

Description:

Adversaries use obfuscation and encoding techniques to conceal the contents of API payloads from inspection tools. WAFs typically rely on pattern matching (signatures) or anomaly detection (heuristics) to block attacks like SQL Injection (SQLi) or Remote Code Execution (RCE). However, modern APIs support complex serialization formats (JSON, Protobuf, GraphQL) that offer multiple ways to represent the same data. Attackers exploit this flexibility to create payloads that break WAF signatures but are normalized and executed by the backend.

Sub-techniques (API Adapted):

● **T1027.00X** - JSON Pollution & Normalization Evasion: Injecting duplicate keys, manipulating content types, or utilizing JSON specific syntax (like unicode escapes) to confuse WAF parsers.
● Using non-standard whitespace, unicode variations, aliases, or double-encoding to break WAF regex signatures targeting GraphQL keywords.
● **T1027.00Z** - gRPC Unknown Field Injection: Utilizing the "Unknown Fields" feature in Protocol Buffers to hide malicious data in fields the WAF

cannot decode or is configured to ignore, which are then processed by the backend or forwarded to vulnerable downstream services.

Protocol-Specific Implementation:

● **REST**: In REST APIs, WAFs often rely on the Content-Type header to determine how to parse the body. A common evasion technique involves JSON Parameter Pollution. Different backend frameworks handle duplicate JSON keys differently. For instance, some parsers (like Python's json library) might accept the last occurrence of a key ("last-one-wins"), while others (like Go's encoding/json or some WAFs) might inspect the first ("first-one-wins").
 ○ Attack Scenario: An attacker sends {"id": "1 OR 1=1", "id": "1"}. A WAF configured to inspect the first occurrence of "id" sees a malicious SQLi payload and blocks it. However, the attacker reverses the payload to {"id": "1", "id": "1 OR 1=1"}. If the WAF only checks the first key (believing it to be the valid one), it permits the request. If the backend application utilizes a "last-one-wins" parser, it processes the second, malicious value, executing the injection.
○ Content-Type Spoofing: Attackers may send a payload with Content-Type: multipart/form-data but format the body as JSON. If the WAF logic is tied strictly to the header, it may attempt to parse the body as a file upload (and find nothing suspicious), while a lenient backend parser (like many Node.js frameworks) might still "sniff" the JSON content and execute it. Recent research identified over 1200 bypasses across major WAFs using these parsing discrepancies.
● **GraphQL**: GraphQL queries are essentially strings sent within a JSON body or as a query parameter. WAFs often use Regular Expressions (Regex) to detect malicious patterns like the introspection query schema.
 ○ Whitespace Evasion: GraphQL supports various characters as separators (commas, spaces, tabs, newlines, and ignored characters). An attacker can inject non-printing characters or excessive whitespace between keywords. For example, transforming query{ schema} into query%0A{%09 schema}. Simple WAF regexes that assume single spaces

or specific delimiters will fail to match, allowing the introspection query to pass. Adobe Commerce, for instance, explicitly documents WAF bypass issues related to GraphQL's repetitive character nature
  ○ Aliases: Attackers use aliases to rename dangerous fields. A WAF looking for systemCmd or users will miss myAlias: systemCmd because the string pattern does not match. The GraphQL engine resolves the alias to the underlying field, executing the restricted operation.
○ Double Encoding: Attackers can double URL-encode characters within the GraphQL query string %255f%255f%2573%2563%2568%2565%256d%2561 (e.g., for schema). If the WAF decodes only once but the backend middleware decodes recursively until it finds a valid string, the payload bypasses inspection.
● **gRPC**: gRPC uses Protocol Buffers (Protobuf), a binary serialization format. WAFs must decode this binary stream to inspect fields, which is computationally expensive and complex.
○ Unknown Fields Evasion: Protobuf allows messages to contain fields that are not defined in the current schema. These are treated as "Unknown Fields." An attacker can inject a malicious payload into a field ID (e.g., field ID 999) that the WAF's schema definition does not recognize. Standard WAF behavior often dictates ignoring unknown fields to prevent false positives during schema updates. However, if the backend application uses a generic deserializer or has a newer schema version that does recognize field 999, the malicious input is processed. This effectively creates a covert channel for exploitation.
  ○ HTTP/2 Multiplexing: gRPC operates over HTTP/2, which supports multiplexing multiple streams over a single TCP connection. Some legacy WAFs struggle to inspect individual frames within a multiplexed stream, or they only inspect the first few frames. Attackers can interleave malicious frames deep within a stream of benign traffic, potentially bypassing flow based inspection logic.

Real-World Example:

Claroty Team82 SQLi Bypass via JSON (2022) Team82 discovered a generic WAF bypass affecting major vendors including Palo Alto Networks, AWS, Cloudflare, F5, and Imperva. The researchers found that many WAFs failed to parse JSON syntax within SQL injection payloads. By prepending JSON-specific syntax to a SQL query (e.g., ' OR JSON_LENGTH('{}')<1 --), the WAFs failed to recognize the SQL injection pattern because they did not support JSON syntax in their SQLi signatures. The backend database engines (which have supported JSON for years) successfully parsed the command and executed the injection. This demonstrated a widespread failure in parsing synchronization between security controls and database engines, allowing attackers to blind the WAF to the attack.

Detection Methods:

● **Discrepancy Analysis:** Implement "canonicalization" where the WAF normalizes inputs (decodes hex, unicode, unescapes JSON) before inspection. Alert on requests where the raw input significantly differs from the normalized input.
● **Strict Content-Type Validation:** Ensure that the API Gateway rejects requests where the Content-Type header does not match the body format.
● **Schema Enforcement:** For gRPC, enforce strict schema validation at the gateway using tools like protovalidate. Reject any request containing unknown fields. For GraphQL, parse the AST (Abstract Syntax Tree) to normalize queries before inspection, rendering whitespace evasion useless.
● **Anomaly Detection in Headers:** Monitor for mismatches between headers and body content, such as JSON content in a multipart/form data request, which is a hallmark of parsing evasion attempts.

Mitigation strategies:

Mitigation	Implementation	Priority
Strict Content-Type	Configure API	Gateways to Critical
Validation	reject requests where the Content-Type header does not match the body format. Disallow multipart/form data on endpoints expecting application/json.	
Drop Unknown Fields (gRPC)	Configure the Protobuf unmarshaller on the server side to strictly DiscardUnknown: true. This prevents the processing of hidden fields that bypass the WAF.	high
GraphQL Query Normalization	Use a specialized GraphQL gateway (e.g., Apollo Router, NGINX App Protect) that parses and normalizes queries before WAF inspection, removing whitespace variations and resolving aliases.	high
Deep JSON Inspection	Ensure WAFs are configured to parse nested JSON depths and handle duplicate keys according to the specific backend behavior (e.g., "last-one-wins" vs	high

	"first-one-wins").	
Input Sanitization	Use parameterized queries and input validation libraries (e.g., protovalidate for gRPC) to ensure that even if a payload bypasses the WAF, it cannot execute.	critical

Technique 2: Hide Artifacts (T1564) - Adapted to "Shadow & Zombie API Usage"

Description:

Adversaries actively seek out and utilize Shadow APIs (undocumented endpoints) and Zombie APIs (deprecated but active endpoints) to evade detection. These endpoints often lack the robust logging, rate limiting, and authentication controls applied to the primary, documented API surface. By directing attacks against these "invisible" assets, adversaries can operate without triggering the alerts associated with the main production environment. This technique leverages the organizational failure to maintain an accurate inventory of API assets and the common practice of leaving legacy endpoints active for backward compatibility.

Sub-techniques (API Adapted):

● **T1564.00X** - Shadow API Exploitation: Targeting undocumented endpoints created for testing, staging, or internal use that are inadvertently exposed to the internet. These often exist outside the API Gateway's routing rules.

● **T1564.00Y** - Zombie API Resurrection: Targeting deprecated versions (e.g., /v1/) of an API that remain active alongside newer, secure versions (e.g., /v2/). Attackers use these to bypass new security controls like MFA or rate limiting.

Protocol-Specific Implementation:

● **REST**: REST APIs frequently version their endpoints in the URL path (e.g., /api/v1/login, /api/v2/login). When upgrading security (e.g., adding MFA or fixing a BOLA vulnerability), developers release v2 but often fail to disable v1 to maintain backward compatibility for legacy clients or mobile apps that haven't updated. ○ Execution: An attacker enumerates the API and discovers v1 is still active. Even if v2 enforces strict rate limiting and MFA, v1 may use simple basic auth and have no rate limits. The attacker directs their brute-force or credential stuffing campaign against v1 completely bypassing the security controls implemented in v2. The security team, monitoring v2, sees no anomalies.
● **GraphQL**: In GraphQL, there is typically a single endpoint (/graphql). However, "Shadow" functionality exists in the form of undocumented mutations or queries that are not removed from the schema but are no longer used by the frontend client.
○ Execution: An attacker uses introspection to find a mutation like updateUserLegacy or debugCreateUser. Since the frontend no longer calls this, the security team may have removed monitoring alerts for it or assumed it was disabled. The attacker uses this mutation to modify user data, bypassing the validation logic present in the current updateUserProfile mutation.
● **gRPC**: Shadow APIs in gRPC often manifest as internal microservices that are accidentally exposed via the ingress controller or load balancer.
○ Execution: A developer configures a gRPC service for internal logging, intended only for service-to-service communication within the cluster. Due to a misconfigured Kubernetes Ingress or API Gateway (e.g., binding to 0.0.0.0 instead of localhost), this service becomes reachable externally. Because it was designed for internal trust, it lacks authentication (or relies

on mTLS which is not enforced at the ingress). An attacker discovers it via port scanning or certificate transparency logs and interacts directly with the internal logic, bypassing the edge security controls.

Real-World Example:

Optus Data Breach (2022) via Zombie API The massive data breach at Australian telco Optus, which exposed 10 million records, was attributed to a "Zombie" API endpoint. The endpoint api.optus.com.au (specifically a legacy endpoint api.optus.com.au/target-api) was intended for internal use or testing and was left publicly accessible without authentication. While the main production APIs were secured, this shadow asset allowed unauthenticated enumeration of customer IDs. The attackers discovered this unmonitored endpoint and exfiltrated data for weeks without triggering the alarms that would have sounded on the main API gateway. This perfectly illustrates how shadow assets neutralize defense-in-depth strategies and allow massive data exfiltration to occur "silently".

Detection Methods:

● **Traffic Baseline Analysis**: Compare live API traffic against the OpenAPI/Swagger specification. Alert on any traffic hitting paths or using parameters that are not defined in the specification (Shadow/Zombie traffic). Use observability platforms to detect "drift" between documentation and reality.
● **Version Usage Monitoring**: Monitor the ratio of traffic between API versions. A sudden spike in traffic to an older version (e.g., /v1/) that typically sees low volume is a strong indicator of an adversary attempting to bypass new security controls.
● **Logs from Edge vs. Gateway:** Correlate logs from the external load balancer with the API Gateway. Traffic that appears at the load balancer but bypasses the API Gateway authentication checks indicates unauthorized direct access to backend services.

● **Ingress Scanning:** Regularly scan public IP ranges for exposed ports (e.g., 50051 for gRPC) that should be internal-only.

Mitigation strategies:

Mitigation	Implementation	Priority
Automated Inventory Discovery	Deploy "sensor" or "agent" based tools (e.g., eBPF, API Security Platforms) that continuously scan the runtime environment to discover all active API endpoints and map them against documentation.	Critical
Strict Deprecation Policy	reject requests where the Content-Type header does not match the body format. Disallow multipart/form data on endpoints expecting application/json.	
Schema Validation Firewall	Configure the WAF/Gateway to block any request that does not match the published OpenAPI/GraphQL High schema. This effectively "closes the door" on undocumented shadow endpoints.	high

Network Segmentation	Isolate internal gRPC microservices from the public ingress using Kubernetes Network Policies or Service Mesh (Istio) authorization policies to prevent accidental exposure.	medium
Legacy Code Audits	Regularly audit codebases for "dead code" routes and controllers that are no longer used but still reachable.	medium

8.Credential Access (TA0006)

Tactic Objective:

The adversary is trying to steal or forge valid account credentials to gain access to the API.

Tactic Description:

In the API domain, Credential Access moves beyond keylogging or dumping hashes from memory. It focuses on industrial-scale automation against authentication endpoints. Adversaries exploit the stateless nature of APIs to launch high-velocity
Credential Stuffing attacks, validating millions of stolen username/password pairs. They also exploit specific protocol features, like GraphQL's Batching, to amplify their attack speed while staying under rate limits. Furthermore, the tactic includes searching for Leaked Tokens (API Keys, OAuth tokens) in public repositories, logs, and client-side code, where "possession" often equals "access" without further verification.

Tactic ID:

TA0006

Total Techniques:

17

Typical Phase:

Intrusion ATT&CK

Version:

v18 (Adapted for API)

Technique 1: Brute Force (T1110) - Adapted to "Batching & Multiplexing Attacks"

Description:

Adversaries use automated tools to guess passwords or validate stolen credentials. In APIs, this is optimized using Batching (GraphQL) and Multiplexing (HTTP/2 in gRPC). Unlike traditional brute force which sends one HTTP request per guess, these techniques allow an attacker to bundle hundreds or thousands of credential checks into a single network request. This drastic increase in efficiency not only speeds up the attack but often bypasses WAF rate limits that count HTTP requests rather than business logic operations.

Sub-techniques (API Adapted):

● **T1110.00X** - GraphQL Batching/Aliasing: Sending multiple mutations in a single GraphQL document to check multiple credentials simultaneously (Brute Force Amplification).
● **T1110.00Y** - gRPC Stream Stuffing: Keeping a long-lived gRPC stream open to bombard the server with login attempts without TCP/TLS handshake overhead.

Protocol-Specific Implementation:

● **GraphQL:** GraphQL allows clients to send multiple queries in one request using Aliases. This was designed to reduce network round- trips but is abused for attacks.

○ Execution: An attacker crafts a single POST request containing 1,000 aliased login mutations.mutation {attempt1: login(user: "admin", pass: "123456") { token } attempt2: login(user: "admin", pass: "password") { token }attempt100: login(user: "admin", pass: "welcome") { token }}
○ Evasion: A rate limiter configured to allow "10 requests per minute" will see this as one request, allowing the attacker to test 1,000 passwords while only consuming 1 unit of their rate limit quota. This technique is known as Brute Force Amplification. It can also be used to bypass 2FA by sending all possible OTP variants in a single batched request.
○ Array-Based Batching: Some GraphQL implementations (like Apollo Server) support sending an array of JSON query objects ([{query:...}, {query:...}]). This achieves the same effect as aliasing but is often easier to automate.
● **gRPC:** gRPC's use of HTTP/2 allows for multiplexing, where multiple requests are sent over a single TCP connection.
○ Execution: An attacker establishes a single authenticated connection (or unauthenticated if the endpoint allows) and opens a Bidirectional Stream. They then push thousands of LoginRequest messages through this stream.Efficiency: Because the connection is persistent, the attacker avoids the latency of the TCP 3-way handshake and TLS negotiation for every attempt. This allows for extremely high-velocity credential stuffing that can overwhelm backend authentication services before network level firewalls react.
○ Evasion: Traditional WAFs that inspect "Requests per Second" (RPS) based on new TCP connections may fail to throttle the high volume of messages inside an established HTTP/2 stream.
● **REST**: While REST lacks batching, attackers utilize Botnets and Residential Proxies (e.g., "Bulletproof Proxies") to distribute attacks. Tools like Sentry MBA, OpenBullet, and Snipr rotate IPs for every request to evade IP-based blocking.
○ Execution: The attacker targets the /api/login endpoint using millions of distinct IP addresses. To the API, it looks like millions of different users logging in once, rather than one attacker logging in a million times. The sheer volume of breached credentials (billions available on the dark web) makes this a highly effective attack despite low success rates per attempt.

Real-World Example:

PayPal Credential Stuffing (2023) In early 2023, PayPal suffered a massive credential stuffing attack where hackers accessed nearly 35,000 accounts. The attackers used a botnet to cycle through valid credentials obtained from other breaches (a technique relying on password reuse). By distributing the traffic across thousands of IPs and mimicking legitimate user agents, they evaded volume-based detection. This incident highlighted that even mature tech giants are vulnerable to high-velocity stuffing if they do not employ advanced behavioral analysis or bot protection. The attackers successfully bypassed standard rate limits by distributing the load, demonstrating the "low and slow" approach at a massive scale.

Detection Methods:

● **Operation-Level Rate Limiting:** For GraphQL, implement "Cost Analysis" or "Depth Limiting." Count the number of resolvers/mutations triggered, not just the number of HTTP requests. Block requests that exceed a certain "complexity cost" or alias count (e.g., max 5 aliases).
● **Behavioral Biometrics:** Analyze mouse movements, keystroke dynamics, and device fingerprinting (Canvas, WebGL) to distinguish between human users and bots. Bots often lack the "jitter" and non-linear movement of human interaction.
● **Impossible Travel (Token Usage)**: Detect if a newly minted token from a successful login is immediately used from a different geolocation, indicating the account was sold or automated.
● **Stream Monitoring (gRPC):** Monitor the number of messages per stream in gRPC. A single stream sending thousands of LoginRequest messages is a clear indicator of stuffing.

Mitigation Strategies:

Mitigation	Implementation	Priority
Query Complexity Limits	In GraphQL, assign a "cost" to the login mutation. Reject queries where the total cost (sum of all aliases) exceeds a threshold (e.g., max 1 login per request).	Critical
Credential Stuffing Protection	Implement a specialized bot mitigation solution (e.g., CAPTCHA, Challenge-Response) that triggers on suspicious login velocity or high failure rates from specific subnets.	High
MFA Enforcement	Enforce Multi-Factor Authentication. Even if stuffing succeeds, the attacker cannot complete the login without the second factor.	Critical
Context-Aware Rate Limiting	Rate limit based on user ID or device fingerprint, not just IP address, to counter distributed botnets.	High
Disable Batching	If not required for business logic, disable batching and aliasing features in the GraphQL server configuration.	Medium

Technique 2: Unsecured Credentials (T1552) - Adapted to "Token Leakage & Hardcoded Secrets"

Description:

Adversaries search for API keys, OAuth tokens, and service account credentials that have been insecurely stored or logged. Unlike searching for passwords in a file system, this involves scanning code repositories (GitHub, GitLab), examining client-side JavaScript, and inspecting public logs. The "bearer" nature of API tokens means possession equates to access; there is often no second factor for machine-to-machine tokens. This technique exploits the "Secret Sprawl" inherent in modern DevOps pipelines.

Sub-techniques (API Adapted):

● **T1552.00X** - Repository Scavenging: Scanning public code commits for hardcoded API keys and secrets using automated tools.
● **T1552.00Y** - Log Traversal: Searching logs (Splunk, ELK) for leaked bearer tokens, PII, or sensitive headers in error messages or debug streams.
● **T1552.00Z** - Client-Side Extraction: Decompiling mobile apps or inspecting browser JS bundles (webpack) to find embedded API secrets intended for backend use.

Protocol-Specific Implementation:

● **REST:** Developers often embed API keys (e.g., AWS_ACCESS_KEY, STRIPE_SECRET, SLACK_BOT_TOKEN) directly into source code for ease of testing. If this code is pushed to a public repository, scanners (like

those used by threat groups such as Lazarus Group) detect them within seconds.
○ Attack: An adversary uses automated tools (e.g., trufflehog, git- secrets, shhgit) to scan GitHub for regex patterns matching standard API key formats (e.g., sk_live_[0-9a-zA-Z]+ for Stripe). Once found, these keys provide immediate, often administrative, access to the API. This bypasses the need for exploitation entirely. If
● **gRPC**: gRPC metadata (headers) often contain authentication tokens. debug logging is enabled (GRPC_TRACE=all or GRPC_VERBOSITY=DEBUG), these headers, including sensitive Bearer tokens, may be written to plain text logs on the server or client.
○ Attack: An attacker with read access to the log server (e.g., via a separate LFI vulnerability or misconfigured S3 bucket) greps for authorization headers to harvest valid session tokens for valid users. They can then replay these tokens to access the gRPC service.
● **GraphQL** : Verbose GraphQL error messages can inadvertently leak configuration data or database credentials if an exception occurs during resolver execution.
○ Attack: An attacker forces an error (e.g., by sending the wrong type of data, like a string to an integer field). The server responds with a stack trace that includes environment variables or connection strings intended only for debugging. This can reveal database passwords or internal API keys.

Real-World Example:

Toyota GitHub Leak (2022) Toyota revealed that an access key to its customer database had been exposed in a public GitHub repository for nearly five years (since 2017). The key allowed access to the T-Connect service, potentially exposing the email addresses and management numbers of nearly 300,000 customers. This incident underscores the "Unsecured Credentials" technique, where the vulnerability is not in the API software itself, but in the operational handling of the secrets used to access it. The key was hardcoded in a script and pushed to a public repo,

remaining undetected by Toyota but potentially visible to any adversary scanning for secrets.

Detection Methods:

● **Secret Scanning**: Implement pre-commit hooks and CI/CD scanning steps that block code commits containing high-entropy strings or known key patterns. Tools like GitGuardian can monitor public repositories for leaks of your organization's keys.
● **Log Auditing**: Configure log aggregation systems (SIEM) to detect and redact patterns resembling API keys or tokens. Alert on any logs that appear to contain Bearer or Authorization strings.
● **Honeytokens**: Deploy fake API keys (canary tokens) in repositories and configuration files. If these keys are ever used against your API, it triggers a high-fidelity alert indicating a repository breach or insider threat.

Mitigation Strategies:

Mitigation	Implementation	Priority
Secrets Management	Use dedicated secrets management vaults (e.g., HashiCorp Vault, AWS Secrets Manager). Inject secrets at runtime via environment variables; never hardcode them.	Critical
Token Rotation	Implement automated, short-lived token rotation. If a key is leaked, its window of validity should be minimal (e.g., 15	High

	minutes to 1 hour). Use refresh tokens for long-term	
Log Sanitization	Configure logging libraries to mask sensitive headers (Authorization, Cookie, Set-Cookie) and scrub PII from error messages before writing to disk.	High
Client-Side Hygiene	Never embed private API keys in mobile apps or frontend JavaScript. Use a "Backend-for-Frontend " (BFF) pattern to proxy requests and keep secrets server-side.	High

9.Discovery (TA0007)

Tactic Objective:

The adversary is trying to figure out the API's structure, available endpoints, data schema, and underlying technology stack to plan further attacks.

Tactic Description:

In an API environment, Discovery is far more structured and "helpful" to the attacker than traditional network scanning. Protocols like GraphQL and gRPC have built-in "self-documentation" features (Introspection and Reflection) that adversaries abuse to map the attack surface instantly. Instead of guessing URLs (fuzzing), the attacker simply asks the API for a map of everything it can do. This tactic also involves analyzing error messages and timing responses to enumerate valid resources (users, tenants, IDs), turning the API's own helpfulness against itself.

Tactic ID:

TA0007

Total Techniques:

34

Typical Phase:

Reconnaissance / Intrusion ATT&CK

Version:

v18 (Adapted for API)

Technique 1: System Information Discovery (T1082) - Adapted to "Schema Introspection & Reflection"

Description:

Adversaries exploit built-in documentation features to obtain a complete blueprint of the API. This includes all available queries, mutations, object types, and arguments. This information transforms a "black box" assessment into a "white box" one, allowing the attacker to identify sensitive or administrative functions that are not publicly documented or linked in the UI.

Sub-techniques (API Adapted):

● **T1082.00X** - GraphQL Introspection Abuse: Querying the schema field to download the full API definition, including hidden types and deprecated fields.
● **T1082.00Y** - gRPC Server Reflection: Using the reflection service to list all available gRPC services, methods, and message formats.
● **T1082.00Z** - OpenAPI/Swagger Exposure: Accessing unauthenticated /docs, /swagger.json, or /openapi.yaml endpoints to download the REST API specification.

Protocol-Specific Implementation:

● **GraphQL**: GraphQL includes a meta-field called schema that allows clients to ask the server about its own structure.
○ Execution: An attacker sends the following query: query {schema { types {namefields { name args { name type { name } } }}}}

○ Result: The server responds with a JSON object detailing every data type (e.g., User, Payment) and operation (e.g., deleteUser, makeAdmin). Even if the frontend UI never uses makeAdmin, the introspection result reveals its existence and the arguments it requires, enabling the attacker to target it directly. Tools like InQL or GraphQL Voyager can visualize this schema, making it easy to spot administrative entry points.
● **gRPC**: gRPC supports a service called grpc.reflection.v1alpha.ServerReflection. If enabled (often by default in dev/staging), it allows tools like grpcurl or grpcui to query the server for its services.
○ Execution: The attacker runs grpcurl -plaintext target.com:443 list.
○ Result: The server returns a list of services, e.g., com.example.InternalAdminService. The attacker can then run describe to see the methods (e.g., ResetDb) and message formats. This exposes internal microservices that may not have been intended for public discovery. Attackers can use this to reverse engineer the proto files without ever seeing the source code.
● **REST:** Developers often deploy auto-generated documentation (Swagger UI) for convenience.
○ Execution: The attacker navigates to standard paths like /v2/api-docs, /swagger-ui.html, or /openapi.json.
○ Result: A full interactive console is presented, allowing the attacker to see every endpoint, expected parameters, and even test requests directly from the browser. This often includes internal endpoints that were not meant to be exposed to the public internet.

Real-World Example:

Peloton API Discovery (2021) Security researcher Jan Masters discovered that Peloton's API allowed unauthenticated users to query data. Crucially, the discovery was aided by the fact that the API endpoint structures were predictable and introspection-like features (or lack of obfuscation) allowed him to map out the user_id fields. He found that he could request private account data (age, gender, city, workout statistics) for

any user, even those with "private" profiles, simply by discovering the correct endpoints and iterating through IDs. This highlights how ease of discovery via predictable schemas leads directly to data exposure (BOLA).

Detection Methods:

● **Introspection Monitoring**: Alert on any GraphQL query containing schema or type originating from external IPs. These should be rare in production traffic.
● **Reflection Logging**: Log access to the gRPC service grpc.reflection.v1alpha.ServerReflection. Any external access to this service is highly suspicious.
● **Honeypot Endpoints:** Create "fake" admin endpoints in your schema/documentation (e.g., mutation deleteProductionDB). If anyone attempts to query them, it is a confirmed positive for reconnaissance, as no legitimate client would know to call them.

Mitigation Strategies:

Mitigation	Implementation	Priority
Disable Introspection/Reflection	Disable GraphQL introspection and gRPC reflection in production environments by default. Only enable them in staging/dev.	Critical
Schema Pruning	If introspection is required (e.g., for public APIs), use tools like graphql-disable introspection or custom middleware to filter out	High

	administrative or sensitive fields from the schema definition returned to public users.	
Obfuscation	Do not use descriptive names for sensitive internal endpoints if they must be exposed. Use UUIDs or opaque identifiers where possible (though security through obscurity is not a primary defense).	Low
Access Control on Docs	Ensure Swagger/OpenAPI documentation endpoints are protected behind authentication, just like the API itself.	Medium

Technique 2: Application Window Discovery (T1010) - Adapted to "Error Message & Stack Trace Fingerprinting"

Description:

Adversaries intentionally send malformed requests to provoke errors. They analyze the resulting error messages, status codes, and response times to "fingerprint" the backend technology stack (identifying frameworks, databases, and versions) and to enumerate valid user accounts (User

Enumeration). This "Chatty Error" vulnerability turns the API into an oracle that leaks system information.

Sub-techniques (API Adapted):

● **T1010.00X** - Stack Trace Analysis: Triggering unhandled exceptions to reveal stack traces (e.g., Java, Python, Node.js) that disclose file paths, library versions, and code structure.
● **T1010.00Y** - User Enumeration via Timing/Errors: Identifying valid accounts based on the time taken to respond or specific error messages (e.g., "User not found" vs "Incorrect password").

Protocol-Specific Implementation:

● **REST:** ○ Error Message Discrepancy: If a login endpoint returns "User does not exist" for one email and "Invalid password" for another, the attacker can build a list of valid users to target with phishing or password spraying.
○ Timing Attack: Even if the error message is generic ("Invalid credentials"), the backend might take longer to process a valid user (due to the computational cost of hashing the password) than an invalid user (failed database lookup). An attacker measures this difference (e.g., 200ms vs 50ms) to enumerate accounts statistically.
● **gRPC:** gRPC has specific status codes that can leak information if not handled carefully.
○ Execution: Sending a request for a user ID. If
○ Analysis: If the server returns NOT_FOUND (5), the user doesn't exist. it returns PERMISSION_DENIED (7) or UNAUTHENTICATED (16), the user likely does exist, but the attacker needs credentials. This subtle distinction allows for enumeration. The gRPC documentation even notes that PERMISSION_DENIED should not be used if the caller cannot be identified, to prevent this leakage, but developers often miss this nuance.

Real-World Example:

GitLab User Enumeration (2022) GitLab faced an issue where their API responses allowed for user enumeration. By querying the API, attackers could distinguish between valid and invalid usernames based on the error response or the presence of public user data. This allowed attackers to compile lists of valid targets for subsequent password spraying attacks. The vulnerability was a classic "oracle" where the API revealed too much information about the existence of a resource, facilitating the Discovery phase of the attack.

Detection Methods:

● Error Rate Monitoring: Alert on spikes in 4xx/5xx errors (REST) or non-zero status codes (gRPC) from a single IP. A high volume of errors often indicates fuzzing or enumeration attempts.
● Response Timing Analysis: Monitor for consistent patterns in response times that align with enumeration scripts (e.g., thousands of requests with identical timing characteristics).
● Output Sanitization Checks: Regularly scan API responses for patterns resembling stack traces or internal IP addresses using DAST tools.

Mitigation Strategies:

Mitigation	Implementation	Priority
Generic Error Messages	Configure the global exception handler to return generic messages (e.g., "Invalid Request") and suppress stack traces in production. Log the	High

	detailed error internally but never return it to the client.	
Consistent Auth Response	Ensure authentication endpoints return the exact same response (content and timing) for both invalid users and invalid passwords to	High
	prevent enumeration. Use bcrypt or argon2 with consistent work factors, or introduce random delays to mask timing differences.	
Rate Limiting on Errors	Implement stricter rate limits for requests that result in errors. If an IP generates 10 errors in a minute block it , temporarily.	Medium

Protocol-Specific Considerations

Each protocol introduces unique attack surfaces that require tailored defenses. Understanding these nuances is critical for adapting the MITRE framework effectively.
Threat Actor Case Studies
Lazarus Group (Hidden Cobra)
● Profile: A North Korean state-sponsored group known for financial theft and espionage.
● API TTPs: Lazarus is known for Defense Evasion via custom packing and "DLL Side-Loading" to hide API interactions within legitimate processes. They frequently use Discovery tactics involving extensive scanning of public facing servers for specific software versions (e.g., Log4j)

to exploit known vulnerabilities. In their attacks on cryptocurrency exchanges, they have utilized Credential Access techniques involving sophisticated phishing to steal API keys and 2FA tokens directly from victims' devices. They have also been observed using custom gRPC-based Command and Control (C2) channels to evade detection, leveraging the protocol's binary nature to blend in with legitimate traffic.
● Relevance: Their ability to blend API exploitation with system level evasion (rootkits, custom protocols) makes them a premier threat to high-value API targets.

APT29 (Cozy Bear / Midnight Blizzard)
● Profile: A Russian state-sponsored group associated with the SVR.
● API TTPs: APT29 specializes in Credential Access via "Token Theft" and "Golden SAML" attacks. They abuse the trust relationships in API authentication protocols (OAuth, SAML) to forge valid credentials, effectively bypassing MFA. For Discovery, they are known to map out Azure AD and Microsoft Graph API permissions extensively to identify high-value targets without triggering alarms. Their use of "Residential Proxies" for Defense Evasion during password spraying campaigns complicates IP-based blocking. They effectively turn the identity provider (IdP) against the organization.
● Relevance: They demonstrate that the identity layer (tokens, cookies) is the new perimeter. Defending against them requires rigorous monitoring of token issuance and usage anomalies.
Detection and Response Framework Implementing an effective API defense requires a layered approach that moves beyond simple signature matching.
1. Ingress Analysis (WAF & Gateway):
○ Normalization: Decode all inputs (URL, Hex, Unicode, JSON escapes) before inspection to counter obfuscation.
○ Schema Validation: Enforce strict adherence to OpenAPI/GraphQL schemas. Drop unknown fields in gRPC.
○ Protocol Hygiene: Verify Content-Type matches the body and enforce method restrictions.
2. Behavioral Analytics (Runtime):
○ Volume vs. Value: Detect "low and slow" attacks where request volume is low but business impact (e.g., 1,000 aliased logins) is high.
○ Sequence Analysis: Alert on illogical API sequences (e.g., Introspection followed immediately by a Mutation, or a sudden burst of errors followed by a successful login).

○ Identity Anomaly: Correlate token usage with geolocation and device fingerprinting to detect stuffing and token theft (Impossible Travel).
3. Honeypots & Deception:
○ Deploy "Shadow" API endpoints (e.g., /api/admin/backup) that serve no business purpose. Any interaction with them (Discovery) or attempt to use keys found within them (Credential Access) triggers an immediate high-fidelity alert.

Mitigation Best Practices

To robustly defend against Defense Evasion, Credential Access, and Discovery in API environments, organizations must adopt the following best practices:
1. Zero Trust for APIs: Authenticate and Authorize every request. Never assume an internal gRPC service is safe from external access. Use mTLS where possible.
2. Strict Schema Enforcement: Use the API schema (OpenAPI, GraphQL, Proto) as a firewall. If a request does not strictly conform to the schema (including unknown fields), block it.
3. Unified Visibility: Maintain a real-time, automated inventory of all API assets (including Shadow and Zombie APIs) to eliminate blind spots. You cannot protect what you cannot see.
4. Cost-Based Rate Limiting: Move beyond "Requests per Second." Implement rate limits based on query complexity (GraphQL) or stream volume (gRPC) to stop amplification attacks.
5. Data Minimization: Ensure error messages are generic (no stack traces) and introspection/reflection features are disabled in production to deny adversaries easy discovery paths.
6. Secrets Management: Eliminate hardcoded secrets. Use vaults and automated scanning to prevent credential leakage in code and logs.
By implementing these strategies, organizations can significantly raise the cost of attack for adversaries, forcing them out of the "silent phase" and into noisy behaviors that are easier to detect and block.

10. Lateral Movement (TA0008)

Tactic Objective:

The adversary attempts to move through the API environment, pivoting from one compromised component—such as a container, a service account, or a user session—to another to gain access to additional resources or privileges.

Tactic Description:

In the context of APIs, "Lateral Movement" is fundamentally different from traditional network traversal. It rarely involves scanning subnets for open ports in the conventional sense. Instead, it is characterized by Identity Pivoting and Infrastructure Traversal. Adversaries exploit the trust relationships inherent in microservices architectures, where Service A is implicitly trusted by Service B, or abuse the identity/access management (IAM) roles attached to compute resources to assume new privileges in the cloud control plane.2 The goal is to transform a compromise of a low-value edge service into control over critical backend databases or cloud infrastructure. This movement often leverages the interconnected nature of cloud APIs, where a single compromised token can unlock a cascade of permissions across the environment.

Tactic ID:

TA0008

Total Techniques:

9

Typical Phase:

Expansion / Deepening Access ATT&CK

Version:
v18 (Adapted for API)

Technique 1: Exploitation of Remote Services (T1210) – Adapted to "Internal Microservice Traversal via SSRF"

Description:

Adversaries exploit Server-Side Request Forgery (SSRF) vulnerabilities to coerce a public-facing API into making requests to internal, non-routable microservices. In modern architectures, internal services often operate with relaxed authentication requirements, assuming that the network perimeter or the API Gateway handles security. By leveraging an SSRF flaw, an attacker effectively bypasses the gateway, interacting directly with backend logic, metadata services, or databases that are otherwise inaccessible from the public internet.6 This technique transforms the vulnerable API into a proxy, allowing the attacker to "surf" the internal network.

Sub-techniques (API Adapted):

• **T1210.00X** – Metadata Service Harvesting: Targeting cloud instance metadata services (IMDS) to steal temporary credentials.
• **T1210.00Y** – Internal API Port Scanning: Using response times and error codes to map internal microservices.
• **T1210.00Z** – Protocol Smuggling: Using schemes like gopher:// or dict:// to communicate with non-HTTP internal services (e.g., Redis, SMTP).

Protocol-Specific Implementation:

REST: The most common and devastating manifestation of this technique involves targeting the cloud provider's metadata service. Most cloud instances (EC2, GCE, Azure VMs) have a local metadata service accessible at a specific link-local address, commonly http://169.254.169.254. This service holds sensitive configuration data, including temporary IAM credentials for the role assigned to the instance.

In a typical attack scenario, an attacker identifies an endpoint that takes a URL as input, such as a profile picture upload feature or a webhook configuration setting. They submit a payload targeting the metadata service, for example, http://169.254.169.254/latest/meta-data/iam/security-credentials/. If the application is vulnerable and lacks proper egress filtering, it will fetch the URL and return the response to the attacker. This response contains the AWS Access Key, Secret Key, and Session Token. The attacker then uses these credentials to authenticate directly to the cloud control plane from their own machine, effectively moving laterally from the application layer to the infrastructure layer. This bypasses firewalls and network segmentation because the request originates from the trusted internal server itself.
gRPC: Microservices often communicate via gRPC over HTTP/2. While standard HTTP SSRF payloads might fail against binary gRPC services due to protocol mismatches, attackers can use "Protocol Smuggling" or specialized payloads if the internal gRPC server supports h2c (HTTP/2 Cleartext) or has a REST gateway enabled.
An attacker can exploit a frontend REST API vulnerable to SSRF to hit internal gRPC endpoints. By crafting a request that hits the internal gRPC port (e.g., 50051), the attacker can interact with internal methods that were never meant to be exposed, such as administrative functions like DeleteUser or AdminReset. Research has shown that attackers can even use h2c upgrades to smuggle commands to backend services, effectively turning a simple SSRF into a mechanism for remote code execution or data manipulation on internal microservices.12 The complexity of gRPC binary payloads often means they are overlooked by standard WAF rules that are tuned for HTTP/1.1 traffic, allowing these internal probes to pass undetected.

Real-World Example:

Capital One Breach (2019)
The Capital One breach serves as the paradigmatic example of this technique and highlights the catastrophic potential of SSRF when combined with overly permissive cloud roles. The attacker, Paige Thompson, a former AWS engineer, exploited an SSRF vulnerability in a misconfigured Web Application Firewall (WAF) hosted on AWS. The WAF, intended to protect the environment, was itself vulnerable. By manipulating the WAF into sending a request to the AWS metadata service

(169.254.169.254), she obtained the temporary IAM credentials assigned to the WAF's EC2 instance.
Critically, the IAM role assigned to this WAF had excessive permissions—specifically, the ability to list and read from S3 buckets that stored sensitive credit card application data for over 100 million customers. Thompson used these stolen credentials to execute API calls to S3, bypassing the application entirely and exfiltrating data directly from the storage plane.14 This incident underscores how a single API vulnerability (SSRF) combined with a failure in least-privilege configuration (IAM) enables lateral movement from a web server to the cloud storage plane, completely circumventing traditional network perimeters.

Detection Methods:

• **Egress Traffic Analysis:** Security teams must monitor for outbound HTTP requests from web servers to private IP ranges (RFC 1918) or link-local addresses, specifically 169.254.169.254. Any traffic to these destinations from an application container should be treated as highly suspicious.
• **Header Anomaly Detection:** Detect requests where the target URL provided in the payload matches internal naming conventions (e.g., internal-billing.local or localhost).
• **Response Time Analysis:** An abnormally fast response can indicate a local network hit (loopback), while a specific timeout might indicate a firewall drop. These timing discrepancies allow attackers to map the internal network even in "blind" SSRF scenarios.

Mitigation Strategies:

• **Input Validation & Allowlists**: The most effective defense is to strictly validate all user-supplied URLs against an allowlist of permitted domains. Do not rely on blacklists, as they are easily bypassed with various encoding techniques.
• **Disable IMDSv1:** Enforce the use of IMDSv2 (Instance Metadata Service Version 2) on AWS. IMDSv2 requires a session token header (X-aws-ec2 metadata-token) that must be retrieved via a PUT request. This simple change makes it significantly harder for an attacker to forge the request via a standard GET-based SSRF exploit.

• **Network Segmentation:** Prevent frontend web servers from routing traffic to sensitive internal management interfaces or data stores except where explicitly required. Use network policies in Kubernetes to restrict pod-to-pod communication.

Technique 2: Cloud Service Dashboard (T1538) – Adapted to "Cloud API Pivoting"

Description:

Once an adversary obtains credentials (keys, tokens) via techniques like SSRF or finding hardcoded secrets, they use these credentials to pivot access across the cloud environment. This represents "Lateral Movement" at the control plane level. Instead of moving from Server A to Server B via SSH, the adversary moves from Identity A to Identity B via API calls like sts:AssumeRole or iam:PassRole. This form of movement is insidious because it often utilizes legitimate API functionality designed for automation and delegation.

Sub-techniques (API Adapted):

• **T1538.00X** – Role Assumption (Role Chaining): Using sts:AssumeRole to trade low-privilege credentials for higher-privilege ones, often moving between accounts or environments.
• **T1538.00Y** – Resource Passing: Using iam:PassRole to assign a privileged role to a compromised compute resource (like a Lambda function or EC2 instance) and then accessing that resource to inherit its powers.
Protocol-Specific Implementation:
AWS API (The PassRole Trap):The iam:PassRole permission is a critical pivot point in AWS environments. It is frequently granted to developers and service accounts to allow them to launch resources like EC2 instances or Lambda functions with specific IAM roles. An attacker who compromises an API key with iam:PassRole and lambda:CreateFunction permissions can execute a privilege escalation attack that functions as lateral movement. The attacker creates a malicious Lambda function—for example, a script

that creates a new admin user or exfiltrates database snapshots—and "passes" an existing high-privilege role (e.g., AdministratorAccess) to this function. When the Lambda executes, it runs with the full permissions of the passed role, effectively elevating the attacker's access.
The attacker has moved laterally from a developer's limited identity to a system administrator's identity, granting them control over the entire AWS account.
Kubernetes API (Token Mounting):In Kubernetes environments, every Pod is mounted with a service account token by default, typically located at /var/run/secrets/kubernetes.io/serviceaccount/token. This token allows the application running in the container to authenticate to the Kubernetes API server.
An attacker who compromises a single container via Remote Code Execution (RCE) can read this token. They then use kubectl (or direct curl requests) to authenticate to the API server. If the service account is over-privileged—for instance, if it is bound to a cluster-admin ClusterRole or has permissions to list secrets in all namespaces—the attacker can pivot from controlling a single isolated container to controlling the entire cluster. They can list other pods, dump secrets, or deploy new malicious containers to other nodes, effectively moving laterally across the cluster infrastructure.

Real-World Example:

TeamTNT & Siloscape
The malware families TeamTNT and Siloscape specifically target Kubernetes environments to execute this form of lateral movement. Siloscape, for instance, is designed to compromise web servers running in Windows containers. Upon infection, it attempts to escape to the host node and searches for Kubernetes node credentials. It then uses these credentials to query the API server, aiming to deploy malicious pods or harvest secrets from other namespaces. This demonstrates a clear and automated kill chain: Initial Access (Web App) -> Execution (Container) -> Lateral Movement (Kubernetes API) -> Impact (Cluster Takeover).
TeamTNT has similarly been observed using scripts that automatically scan for the kubelet API and attempt to leverage anonymous access or stolen tokens to move between nodes and deploy cryptominers.

Detection Methods:

• CloudTrail/Audit Logs: Security teams must alert on sts:AssumeRole calls originating from unexpected IP addresses or user agents that deviate from the baseline behavior of the service.
• Anomalous API Sequencing: Detect the sequence of CreateFunction followed immediately by InvokeFunction and DeleteFunction. This pattern is indicative of "PassRole" abuse where an attacker creates a temporary resource to execute a privileged action and then covers their tracks.
• K8s Audit Logs: Monitor for access to sensitive endpoints like /api/v1/secrets or /api/v1/pods coming from non-system service accounts. Any service account associated with a frontend web application that attempts to list secrets for the entire cluster should trigger an immediate alert.

Mitigation Strategies:

• Least Privilege: Strictly scope iam:PassRole permissions to specific resources using the Resource field in IAM policies (e.g., Resource: arn:aws:iam::.../role/SpecificRole). Never use Resource: *, as this allows the passing of any role, including administrative ones.
• Disable Automounting: For Kubernetes pods that do not need to communicate with the API server, set automountServiceAccountToken: false in the Pod specification. This removes the credential from the file system, denying the attacker a pivot point.
• Cloud Security Posture Management (CSPM): Implement CSPM tools to continuously scan for over-privileged roles, service accounts with excessive permissions, and misconfigured trust policies that could facilitate lateral movement.

Detection Methods:

• Sidecar Process Monitoring: Security tools should detect any process other than the legitimate Envoy binary attempting to access the certificates mounted in the sidecar volume or communicating on the mesh ports.
• Admin Interface Access: Alert on any access to the Envoy admin port (15000) from within the pod or from external sources. This interface should

be strictly locked down.
• Mesh Traffic Anomalies: Use the service mesh telemetry to detect services communicating with endpoints not defined in the ServiceEntry registry or exhibiting unusual traffic patterns. Mitigation Strategies:
• Strict mTLS: Enforce STRICT mTLS mode in Istio PeerAuthentication policies to prevent unauthenticated lateral movement. This ensures that only services with valid, signed certificates can communicate.
• Admin Port Restriction: Configure Envoy to bind the admin interface to localhost only and restrict access using network policies to prevent unauthorized interaction.
• Principle of Least Privilege (Network): Use Sidecar resources in Istio to explicitly limit the egress traffic a service can initiate. By default, a sidecar might know about all services in the mesh; scoping this configuration prevents a compromised service from connecting to arbitrary targets it has no business reaching.

11. Collection (TA0009)

Tactic Objective:

The adversary is trying to gather data of interest to their goal.

Tactic Description:

Collection techniques identify and gather information—such as sensitive files, internal communications, screenshots, and database records—prior to theft4. This stage is the culmination of discovery, where an attacker identifies which data aligns with their primary mission, be it corporate espionage or financial extortion5.

Tactic ID:

TA0009

Total Techniques:

17

Typical Phase:

Post-compromise

Technique 1: Email Collection (T1114)

Description:

Adversaries target email repositories to harvest strategic plans, financial ledgers, and legal correspondence7. Centralized mail servers like Microsoft 365 or on-premise Exchange are high-value targets due to the concentration of cross-departmental data88.

Sub-techniques:

T114.001 - 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.

T1114.002- 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 MailSnipper can be used to automate searches for specific keywords.

T1114.003 - 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.

Real-World Example (2025 Update):

Arctic Tempest (State-Sponsored) was observed targeting a global semiconductor manufacturer[99]. Instead of traditional phishing, they utilized a compromised service principal to grant themselves `Mail.Read` permissions via the Microsoft Graph API. The actors then executed a script to search for attachments containing the string "next-gen-architecture" and

"patent-pending," successfully gathering 15GB of engineering specifications.

Adversaries conduct email collection through a structured progression, moving from gaining access to the mail environment to identifying and extracting specific target data.

Phase 1: Access and Authentication

Before collection can occur, the adversary must establish a foothold in the mail environment. This is typically achieved through one of three primary methods:

- **Credential Theft:** Using valid credentials harvested via phishing, brute force, or credential stuffing to log in to webmail (OWA) or IMAP/SMTP services.
- **Token Abuse (OAuth):** In cloud environments like Microsoft 365 or Google Workspace, adversaries often use malicious apps to trick users into granting OAuth permissions. This allows the attacker to read emails without ever knowing the user's password.
- **Mailbox Permissions:** If an attacker has already escalated privileges to "Domain Admin" or "Global Admin," they may grant themselves "Full Access" permissions to specific high-value mailboxes (e.g., CEO, CFO, or HR).

Phase 2: Environment Discovery

Once inside the mail environment, the adversary needs to map the territory to find the most relevant information:

- **Mailbox Enumeration:** The attacker lists all available mailboxes and identifies key personnel.
- **Permission Mapping:** They check which folders are accessible (Inbox, Sent, Archive, or Shared Mailboxes) and search for existing delegates.

- **keyword Identification:** Attackers identify "hot" keywords used within the organization (e.g., "Project X," "Acquisition," "Invoice," or "Password") to refine their search.

Phase 3: Search and Collection

In this phase, the adversary begins the actual gathering of data. This can be done manually, but advanced actors use automation:

- **Search Queries:** Using built-in search functions (like the `Search-Mailbox` cmdlet in Exchange) to find messages with specific keywords or attachments.
- **Recursive Harvesting:** Scripts are used to walk through every folder in a mailbox and copy all messages to a local or hidden directory.
- **Targeted Extraction:** Instead of taking everything, the attacker may specifically target attachments (PDFs, Excel sheets) which are likely to contain structured data.

Phase 4: Staging and Persistence

To ensure the data is ready for exfiltration without triggering immediate alerts:

- **Email Forwarding Rules:** The adversary creates a "hidden" rule that automatically forwards all incoming or outgoing mail containing specific keywords to an external attacker-controlled address.
- **Local Archiving:** On a compromised endpoint, the attacker may find and copy local archive files (like `.pst` or `.ost` files) which contain years of historical data.
- **Compression/Encryption:** The gathered emails are often compressed into a single encrypted file (e.g., a password-protected `.zip` or `.7z`) to hide the content from Data Loss Prevention (DLP) scanners during the final move.

Advanced Detection Methodologies:

- Monitor for an unusual volume of Microsoft Graph API or Exchange Web Services (EWS) calls from non-administrative service accounts[1111].
- Alert on the creation of hidden inbox rules or auto-forwarding rule modifications (Windows Event ID 4656)[12121212].
- Identify high-frequency mail synchronization events originating from unusual geographic locations or known proxy exit nodes[1313].

Mitigation Strategies:

- **API Permission Hardening:** Restrict OAuth application permissions to "Application Access Policy" scopes to limit what a service principal can see[14].
- **Enforce MFA:** Ensure that all administrative access to the mail environment is protected by hardware-based MFA[15].
- **DLP Integration:** Deploy Data Loss Prevention for email to block the transmission of specific internal-only keywords to external domains[1616].

Technique 2: Automated Collection (T1119)

Description:

Adversaries utilize scripts to scan local systems and network shares for files matching specific extensions or metadata without manual oversight1717.
cloud-based environments, adversaries may also use cloud APIs, data pipelines, command line interfaces, or extract, transform, and load (ETL) services to automatically collect data.
This functionality could also be built into remote access tools.

ID:

T1119

Sub-techniques:

No sub-techniques

Platforms:

IaaS, Linux, Office Suite, SaaS, Windows, macOS

Contributors:

Arun Seelagan, CISA; Praetorian

Ember Dragon (Financial Actor)

deployed a specialized Go-based binary that performed recursive searches across all SMB shares The tool was programmed to identify any file modified within the last 48 hours with a size greater than 1MB, ensuring they only collected the most current and relevant financial data for the organization's upcoming quarterly report.

Detection Strategy

Automated execution of native utilities and scripts to discover, enumerate, and exfiltrate files and clipboard content. Focus is on detecting repeated file access, scripting engine use, and use of command-line utilities commonly leveraged by collection scripts.
Repeated or automated access to user document directories or clipboard using shell scripts or utilities like xclip/pbpaste. Detectable via auditd syscall logs or osquery file events.
Suspicious sign-ins to Graph API or sensitive resources using non-browser scripting agents (e.g., Python, PowerShell), often for programmatic access to mailbox or OneDrive content.

Mitigations

Encryption and off-system storage of sensitive information may be one way to mitigate collection of files, but may not stop an adversary from acquiring the information if an intrusion persists over a long period of time and the adversary is able to discover and access the data through other means. Strong passwords should be used on certain encrypted documents that use them to prevent offline cracking through Brute Force techniques.
Encryption and off-system storage of sensitive information may be one way to mitigate collection of files, but may not stop an adversary from acquiring the information if an intrusion persists over a long period of time and the adversary is able to discover and access the data through other means.

12. Command and Control (TA0011)

Tactic Objective:

The adversary is trying to communicate with compromised systems to control them.

Tactic Description:

Command and Control (C2) represents the methods used to maintain a persistent connection with a victim system21. Modern adversaries favor "Living-off-the-Network" techniques, using legitimate web protocols to blend in with standard corporate traffic22.

Tactic ID:

TA0011

Total Techniques:

18

Technique 1: Application Layer Protocol (T1071)

Description:

The use of standard protocols like HTTP, HTTPS, or DNS to bypass egress filtering and maintain stealth23.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.

ID:

T1071

Subtechniques:

T1071.001- 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.
T1071.003- 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.
T1071.004- 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.
T1071.005-Protocols such as MQTT, XMPP, AMQP, and STOMP use a publish/subscribe design, with message distribution managed by a centralized broker.Publishers categorize their messages by topics, while subscribers receive messages according to their subscribed topics.An adversary may abuse publish/subscribe protocols to communicate with systems under their control from behind a message broker while also mimicking normal, expected traffic.

Real-World Example (2025 Update):

Void Marauder utilized a unique C2 implementation that mimicked the traffic patterns of a popular collaborative music streaming service[2424]. Commands were hidden within the metadata of playlist updates, allowing the malicious traffic to bypass traditional signature-based firewalls that viewed the connection as legitimate employee activity.

The execution of this technique typically follows four distinct phases:

Phase 1: Infrastructure Preparation

Before establishing a connection, the adversary must set up the "listening" end of the communication channel.

- **Domain Acquisition:** Registering domains that look legitimate (typosquatting, e.g., `micros0ft-update.com`) or compromising existing reputable websites to act as proxies.
- **Server Configuration:** Setting up a C2 server (e.g., Cobalt Strike, Sliver, or custom frameworks) to handle incoming requests on standard ports like 80 (HTTP) or 443 (HTTPS).
- **CDN/Proxy Setup:** Often, attackers use "Domain Fronting" or Content Delivery Networks (CDNs) to hide the true IP address of the C2 server behind a trusted provider's infrastructure.

Phase 2: Establishing the Connection (Beaconing)

Once the malware is executed on a victim's machine, it initiates the first outbound "phone home."

- **Protocol Selection:** The malware identifies which protocols are allowed. If HTTPS is open, it will wrap its commands in encrypted web traffic.
- **Beaconing Pattern:** The compromised system sends a "beacon"—a small packet of data—at set intervals (e.g., every 60 seconds) to ask the C2 server for instructions.

- **Jitter:** To avoid detection by automated traffic analyzers that look for perfectly timed heartbeats, attackers add "jitter" (randomized delays) to the beaconing frequency.

Phase 3: Command Exchange and Tunneling

In this phase, the actual "Command" and "Control" occur.

- **Request/Response Cycle:** The victim system sends an HTTP GET request to the server. The server hides instructions (like "execute shell command" or "download file") inside the HTTP response body or headers (e.g., Cookie or User-Agent fields).
- **Data Encapsulation:** Large pieces of data or tools are broken down into small chunks and hidden within standard protocol fields to avoid triggering size-based alerts in Data Loss Prevention (DLP) systems.

Phase 4: Persistence and Stealth

The adversary ensures the channel remains open while avoiding discovery by security analysts.

- **Living-off-the-Network:** The traffic is carefully crafted to look exactly like a browser or a legitimate application (like a Slack or Google Drive API call).
- **Fallback Channels:** Sophisticated malware is programmed with secondary protocols. If HTTPS is blocked, it might switch to DNS Tunneling (T1071.004) or SMB.
- **Dead Drop Resolvers:** Attackers may use legitimate social media profiles or forum posts as "dead drops" to host the IP address of their current C2 server, allowing the malware to update its connection info dynamically.

Advanced Detection Methodologies:

- **Beaconing Analysis:** Analyze network flows for highly rhythmic heartbeat signals that indicate automated C2 check-ins[262626].

- **TLS Fingerprinting:** Use JA3/JA3S fingerprints to identify non-standard TLS client implementations frequently used by custom malware[27].
- **Entropy Checks:** Analyze HTTP POST payloads for high entropy, which typically indicates custom encryption rather than standard web data[28].

Mitigation Strategies:

- **SSL/TLS Inspection:** Decrypt and inspect outbound HTTPS traffic at the perimeter to identify hidden commands[29].
- **DNS Filtering:** Implement a protective DNS service to block resolutions for newly registered domains and known DGA addresses[30].

Technique 2: Data Encoding (T1132)

Description:

Adversaries may encode data 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. 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.

ID:

T1132

Sub-Techniques:

T1132.001-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.
T1132.002-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.

Real-Life Example: The SolarWinds Supply Chain Attack (APT29 / Nobelium)

One of the most sophisticated real-world examples of **T1132.001 (Standard Encoding)** was observed during the **SolarWinds (SUNBURST)** compromise in 2020, attributed to the threat group **APT29** (also known as Nobelium).

Detection Methodologies

Detecting encoding requires moving beyond signature matching and focusing on the statistical properties of data.

- **Entropy Analysis:** Encoded data (especially Base64) usually has significantly higher entropy (randomness) than standard English text. Security tools can be configured to flag network packets or file sections with unusually high entropy.
- **Process Command Line Monitoring:** Monitor for suspicious flags in command lines, such as `-e`, `-enc`, or `-EncodedCommand` in

PowerShell, which indicate that an encoded script is being passed to the interpreter.
- Decoding Utilities: Monitor for the use of built-in system tools like certutil.exe or openssl being used with "decode" or "base64" flags, as these are often used by attackers to "unpack" their payloads once they reach the target.

Mitigation Strategies

- **PowerShell Script Block Logging:** Enable Event ID 4104. This is a critical defense because Windows will log the *decoded* or de-obfuscated version of the script in the event logs just milliseconds before it actually executes.
- **Application Whitelisting:** Restrict the execution of administrative tools like `certutil.exe` to only those users who require them for their jobs.
- **Network Inspection:** Use SSL/TLS inspection to decrypt egress traffic. This allows deep packet inspection (DPI) to identify high-entropy strings within HTTP headers that should normally contain standard text.

13. Exfiltration (TA0010)

Tactic Objective:

The adversary is trying to steal data[31].

Tactic Description:

Exfiltration is the removal of data from the target network32. This is often the final phase of a theft-oriented operation and may occur over the primary C2 channel or through alternative, higher-bandwidth paths33.

Tactic ID:

TA0010

Total Techniques:

9

Technique 1: Exfiltration Over Cloud Storage (T1567)

Description:

Adversaries abuse legitimate cloud services (e.g., OneDrive, Mega, Dropbox) to move data, as these services are often pre-approved by IT departments34.

ID:

T1567.002

Sub-technique of:

T1567

ID: T1567.001

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.

ID: T1567.002
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.

ID: T1567.003
Text storage sites are often used to host malicious code for C2 communication (e.g., Stage_Capabilities), but adversaries may also use these sites to exfiltrate collected data. Furthermore, paid features and encryption options may allow adversaries to conceal and store data more securely.

The execution of this technique follows a structured four-phase lifecycle:

Phase 1: Infrastructure Provisioning

Before the data theft begins, the adversary must establish a destination that they control.

- **Account Creation:** The attacker registers for a free or paid account on a public cloud storage platform. To avoid attribution, they often use stolen credentials or "burner" email addresses.
- **API Key Acquisition:** For automated exfiltration, the attacker generates API keys or OAuth tokens for the account. This allows their malware to upload data directly to the cloud storage without requiring a manual login or a browser interface.

Phase 2: Tooling and Preparation

The adversary identifies or deploys the software necessary to move the data.

- **Living-off-the-Land (LotL) Tools:** Attackers often use legitimate, "dual-use" command-line utilities like **rclone**, **rrsync**, or **AWS CLI**. Because these tools are used by IT professionals for legitimate backups, their presence on a system may not trigger immediate security alerts.
- **Malware Integration:** In some cases, custom malware is programmed with built-in cloud API libraries (e.g., using the Dropbox SDK) to perform the upload natively without needing external tools.

Phase 3: Staging and Obfuscation

To ensure the exfiltration is fast and avoids detection by deep packet inspection (DPI), the data is prepared for transport.

- **Data Bundling:** Stolen files are collected and compressed into large archives (e.g., `.zip` or `.7z`).
- **Encoding and Encryption:** The archives are often encrypted with a password or encoded (T1132) to prevent DLP tools from scanning the contents for sensitive keywords (like "Social Security" or "Confidential") as they leave the network.
- **Hidden Directories:** The staged data is placed in obscure local folders, such as `C:\Windows\Temp\` or deep within the `AppData` directory, to hide it from the user.

Phase 4: Data Synchronization (The Exfiltration)

The final phase is the actual transmission of data from the internal network to the attacker's cloud account.

- **Encrypted Outbound Traffic:** The upload occurs over HTTPS (Port 443). To the network firewall, this looks like a standard user uploading a file to a trusted site.

- **Chunking:** To avoid triggering alerts on "large outbound file transfers," the tools may break the data into small chunks and upload them over several hours or days.
- **Cleanup:** After the synchronization is complete, the adversary often deletes the local staging files and the exfiltration tools to remove traces of the activity.

Real-World Example (2024 Update):

During an attack on a South American telecommunications firm, Neon Sandstorm used the legitimate rclone utility to synchronize 400GB of customer records directly to a private AWS S3 bucket35. By using a tool frequently used by the organization's own backup scripts, the exfiltration blended perfectly with standard operations36.

Advanced Detection Methodologies:

- Monitor for the execution of unauthorized synchronization tools like rclone, rrsync, or unauthorized cloud desktop clients[37].
- Alert on large data transfers to IP ranges belonging to public cloud providers that do not match the organization's known cloud footprint[38].
- Track outbound protocol anomalies where a system suddenly starts communicating via high-bandwidth HTTPS to a previously unknown cloud endpoint[39].

Mitigation Strategies:

- **CASB Enforcement:** Utilize a Cloud Access Security Broker to allow only corporate-sanctioned cloud accounts and block personal storage sites[40].
- **Egress Rate Limiting:** Implement outbound bandwidth throttling to slow down large-scale exfiltration attempts.

Technique 2: Traffic Duplication(T1020)

Description:

Adversaries may leverage traffic mirroring in order to automate data exfiltration over compromised infrastructure. Traffic mirroring is a native feature for some devices, often used for network analysis. For example, devices may be configured to forward network traffic to one or more destinations for analysis by a network analyzer or other monitoring device.

ID:

T1020

Sub-techniques:

T1020.001
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 Alteration Protocol.

Phase 1: Access and Reconnaissance

Before traffic can be duplicated, the adversary must gain the necessary administrative access to the network infrastructure.

- **Infrastructure Targeting:** The attacker identifies high-value network "aggregation points," such as core switches, routers, or virtual distributed switches (vDS), where the most sensitive data (credentials, unencrypted API keys) converges.
- **Privilege Acquisition:** The adversary must compromise an account with administrative privileges on the network device. This is often achieved through **Credential Access (TA0006)** techniques like

brute-forcing SSH or exploiting vulnerabilities in the switch's management interface.

Phase 2: Configuration and Redirection

Once administrative access is established, the adversary configures the device to start the duplication process.

- **SPAN/Mirror Configuration:** On physical switches, the attacker enables **SPAN (Switched Port Analyzer)** or **Remote SPAN (RSPAN)**. They designate a "source" port (the victim's traffic) and a "destination" port (where the duplicate traffic is sent).
- **Encapsulated Mirroring (ERSPAN):** For modern or virtualized environments, attackers may use **ERSPAN**, which encapsulates the mirrored traffic in a GRE tunnel, allowing them to send the duplicated data across the network to a remote IP address they control.
- **Virtual Mirroring:** In cloud environments (AWS/Azure), the attacker may enable "VPC Traffic Mirroring" on a compromised instance's network interface.

Phase 3: Capture and Storage (The "Sniffer" Setup)

The duplicate traffic must be received and processed by a listener, often referred to as a "sniffer."

- **Listener Deployment:** The adversary sets up a workstation or a compromised server with a network interface in "promiscuous mode."
- **Tool Execution:** They run packet capture utilities such as **tcpdump**, **Wireshark**, or **Tshark** to record the incoming stream.
- **Local Staging:** Because network traffic is voluminous, the attacker often uses filters (BPF - Berkeley Packet Filters) to only save "interesting" traffic, such as packets containing strings like `PASS`, `LOGIN`, or `AUTH`, to a local staging file.

Phase 4: Analysis and Credential Harvesting

The final phase involves extracting value from the captured data.

- **Protocol Parsing:** The attacker analyzes the captured PCAP files to reconstruct sessions. They look for unencrypted protocols like HTTP, FTP, Telnet, or LDAP.
- **Session Hijacking:** By capturing active session tokens or cookies from the duplicated traffic, the adversary can perform **Session Hijacking** to gain access to web applications without needing a password.
- **Internal Mapping:** The traffic provides a "blueprint" of the network, revealing internal IP addresses, service versions, and communication patterns that guide the next stages of the attack.

Real World Example:

In many Advanced Persistent Threat (APT) scenarios, **Traffic Duplication** is used as a stealthy alternative to "Man-in-the-Middle" (MitM) attacks. While a MitM attack might be detected by latency or certificate warnings, a duplicated port is invisible to the user, making it a preferred technique for long-term espionage and credential harvesting.

Detection Methodologies

Detection of traffic duplication is challenging because it is inherently passive and does not alter the original packets. Security teams must focus on the configuration changes and the physical/logical "destination" of the mirrored traffic.

1. **Network Device Configuration Auditing:**
    - Monitor for changes to network switch and router configurations, specifically looking for the enabling of **SPAN (Switched Port Analyzer)**, **RSPAN**, or **ERSPAN** (Encapsulated Remote SPAN) ports.

    - Alert on any configuration command that includes keywords like `monitor session`, `destination interface`, or `source interface` on Cisco or similar enterprise hardware.
2. **Unusual Egress Traffic Volume:**
    - Traffic duplication often results in a significant increase in outbound traffic from a single network device. Monitor for internal interfaces that suddenly begin transmitting high volumes of "one-way" traffic to an external or unauthorized internal IP address.
    - Use **NetFlow or IPFIX** to detect long-duration sessions between a network backbone switch and a workstation that does not normally handle administrative traffic.
3. **Host-Based Artifacts:**
    - Monitor for the installation of packet capture drivers or libraries (e.g., **Npcap**, **WinPcap**, or **libpcap**) on systems that do not belong to the network engineering or security teams.
    - Alert on the execution of tools like Wireshark,tcpdump or tcpshark on non-administrative endpoints.

Mitigation Strategies

Mitigation involves hardening the network infrastructure to prevent unauthorized mirroring and ensuring that even if traffic is captured, it remains unreadable.

1. **Strict Administrative Access Control (PAM):**
    - Restrict access to the command-line interface (CLI) of network switches and routers. Only highly audited administrative accounts should have the "Privileged EXEC" mode required to configure port mirroring.
    - Implement **Multi-Factor Authentication (MFA)** for all network infrastructure logins.
2. **Encryption of Data in Transit:**
    - The most effective mitigation is to ensure that all internal traffic is encrypted (e.g., using **IPsec** or **TLS**). If the adversary

successfully duplicates the traffic, they will only capture encrypted payloads that are useless without the private keys.

3. **Physical Port Security:**
    - Disable unused ports on switches and use "MAC-address sticky" features to prevent unauthorized devices (like a "Network Tap" or an attacker's laptop) from being plugged directly into the physical network infrastructure.
4. **Network Segmentation and VLAN Hopping Protection:**
    - Ensure that the management VLAN (where switch configurations occur) is isolated from general user traffic. This prevents an attacker on a standard workstation from reaching the switch management interface to enable mirroring.

14. Impact (TA0040)

Tactic Objective:

The adversary is trying to manipulate, interrupt, or destroy your systems and data[42].

Tactic Description:

Impact techniques represent the "end of the road" for an attacker—actions that result in the disruption of availability or the destruction of data integrity43. While other tactics focus on theft, Impact focuses on damage44.

Tactic ID:

TA0040

Total Techniques:

13

Technique 1: Data Encrypted for Impact (T1486)

Description:

Commonly known as ransomware, adversaries encrypt a victim's data to make it inaccessible, typically followed by a demand for payment.
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

Real-World Example (2024 Update):

Iron Sleet targeted a regional healthcare provider using a sophisticated "living-off-the-hypervisor" attack[46]. They compromised the management console of the virtualization environment and used the built-in storage migration tools to move all active virtual machine disks into an encrypted volume, effectively locking out the entire hospital's digital infrastructure in under 20 minutes[47].

Adversaries typically execute this technique through a structured four-phase lifecycle:

Phase 1: Preparation and Environment Hardening

Before starting the encryption process, the adversary ensures that the environment is "primed" so the attack cannot be interrupted by security software or system administrators.

- **Privilege Escalation:** The attacker gains administrative or SYSTEM-level privileges to ensure they can access all local and network files.
- **Service Termination:** Malicious scripts are used to stop database services (to unlock files like SQL or Oracle databases so they can be encrypted) and security services (to disable antivirus or EDR monitoring).
- **Inhibiting Recovery:** The attacker disables system features that could allow the user to undo the damage, such as Windows Error Recovery or Automatic Repair.

Phase 2: Destruction of Backups (Pre-Encryption)

To ensure the victim has no choice but to pay, the adversary destroys any local or reachable recovery options.

- **Shadow Copy Deletion:** Using native tools like `vssadmin.exe delete shadows /all /quiet`, the attacker removes the local "snapshots" Windows creates for file recovery.

- **Backup Server Targeting:** If the attacker has moved laterally (TA0008), they will log into backup servers to delete cloud syncs, wipe tape libraries, or encrypt the backup repository itself.
- **Boot Configuration Modification:** Using `bcdedit`, they may disable the ability for the computer to boot into safe mode or recovery environments.

Phase 3: The Encryption Engine

The actual transformation of data occurs in this phase. To maximize speed and minimize the chance of detection, attackers use specific cryptographic strategies.

- **File Discovery:** The ransomware scans all local drives, mapped network drives, and attached cloud storage for specific extensions (e.g., `.docx`, `.pdf`, `.sql`, `.zip`).
- **Symmetric/Asymmetric Hybrid:** The malware typically uses a fast symmetric algorithm (like AES) to encrypt the files. It then encrypts that AES key with a public asymmetric key (like RSA-2048) that belongs to the attacker. This means only the attacker's private key can unlock the files.
- **Intermittent Encryption:** To evade EDR tools that monitor for high CPU usage, some modern ransomware only encrypts "chunks" of a file (e.g., every 10MB), making the file unusable while remaining stealthy.

Phase 4: Notification and Persistence (The Ransom Note)

Once the files are encrypted, the adversary makes their presence known.

- **Extension Modification:** Files are renamed with a unique extension (e.g., `.lockbit`, `.blackcat`, or `.crypted`).
- **Ransom Note Deployment:** A text file (e.g., `README.txt`) or an HTML file is placed in every folder and often set as the desktop wallpaper. This note contains instructions on how to access a "leak site" on the dark web and the amount of the ransom.

- **Persistence:** In some cases, the malware remains on the system to re-encrypt any files the user attempts to restore from a non-air-gapped backup.

Advanced Detection Methodologies:

- Monitor for mass file modification events or rapid renames of files on network shares[48].
- Detect the execution of native backup deletion tools such as `vssadmin.exe delete shadows` or `bcdedit /set {default} recoveryenabled No`[494949].
- Alert on high CPU and Disk I/O spikes associated with unknown binaries performing cryptographic operations[50].

Mitigation Strategies:

- **Offline Backups:** Maintain air-gapped, immutable backups that cannot be modified by network-resident ransomware[51].
- **Shadow Copy Protection:** Use endpoint security policies to prevent the unauthorized deletion of Volume Shadow Copies[52].
- **Network Segmentation:** Isolate critical OT/ICS networks from the IT environment to prevent lateral spread of destructive wipers.

Technique 2: Defacement(T1491)

Description:

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.

ID:

T1491

Sub-techniques:

ID: T1491.001- 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 server login messages, or directly to user systems with the replacement of the desktop wallpaper.[1][2] 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.
T1491.002- 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.

The Phases of a Defacement Attack

A successful defacement typically follows three technical phases:

1. **Exploitation:** The attacker identifies a vulnerability in the web server or Content Management System (CMS), such as an unpatched version of WordPress, a SQL injection vulnerability, or compromised administrative credentials.
2. **Modification:** Once access is gained, the adversary replaces the site's `index.html` or other core files with their own malicious

content. They may also modify the database to change text across the entire platform.
3. **Publicity:** The attacker often takes a screenshot and posts it to "mirror" sites (like Zone-H) or social media to prove the successful hack and gain notoriety.

Real World Example:

The "Sandworm" attacks on Ukraine (2022)

A prominent and high-impact example of defacement occurred in January 2022, just prior to the Russian invasion of Ukraine. This attack was attributed to the Russian-linked group Sandworm (APT44).

Adversaries targeted approximately **70 Ukrainian government websites**, including the Ministry of Foreign Affairs and the Ministry of Education.

- **The Content:** The attackers replaced the homepages with a threatening message in Ukrainian, Polish, and Russian, stating: *"Ukrainians! All your personal data was uploaded to the public network... Wait for the worst."*
- **The Strategic Use:** While the defacement was the most visible part of the attack, it served as a **distraction**. Behind the scenes, the attackers were deploying **WhisperGate** wiper malware to destroy data on government servers. The defacement created public panic and overwhelmed incident response teams, allowing the destructive malware more time to function undetected.

Detection Methodologies

- **File Integrity Monitoring (FIM):** This is the most effective detection method. FIM tools monitor the web server's root directory and alert instantly if core files like `index.php` or `header.php` are modified by an unauthorized process.

- **HTTP Response Monitoring:** Monitor for sudden changes in the "size" or "hash" of your homepage response. If your standard homepage is 50KB and suddenly drops to 2KB (the size of a simple defacement page), an alert should trigger.
- **External Website Monitoring:** Use third-party services that "crawl" your site from the outside to verify that the visual content remains unchanged.

Mitigation Strategies

- **Principle of Least Privilege:** Ensure the web server process (e.g., Apache or Nginx) does not have "Write" permissions to its own web root. Content should only be pushed to the server via a secure deployment pipeline (CI/CD) using a separate service account.
- **CMS Hardening:** Regularly patch all plugins and themes. Use a Web Application Firewall (WAF) to block common exploit attempts like SQL Injection (SQLi) and Cross-Site Scripting (XSS).
- **Static Site Generation:** For high-value informational sites, use static site generators. Since there is no database or active code on the server, it is significantly harder for an attacker to modify the content dynamically.

APT Group Case Studies

This section provides in-depth profiles of prominent Advanced Persistent Threat (APT) groups and their usage of MITRE ATT&CK tactics and techniques.

APT29 (Cozy Bear / The Dukes)

Attribution: Russian Foreign Intelligence Service (SVR)
Active Since: 2008
Motivation: Espionage, intelligence gathering
Sophistication Level: Very High

Notable Campaigns:

- SolarWinds Supply Chain Attack (2020)
- Democratic National Committee breach (2016)
- COVID-19 vaccine research targeting (2020)

Tactics, Techniques, and Procedures (TTPs):

Reconnaissance

- T1589.002 – Email Addresses
- T1595.002 – Vulnerability Scanning

Extensive OSINT gathering targeting government employees and contractors

Resource Development

- T1587.001 – Malware Development

Developed custom malware
(SUNBURST, SUNSPOT)

Initial Access

- T1195.002 – Supply Chain Compromise
- T1566.001 – Spearphishing Attachment

SolarWinds Orion software compromise

Execution

- T1059.001 – PowerShell
- T1106 – Native API

Fileless execution, living-off-the-land techniques

Persistence

- T1543.003 – Windows Service
- T1547.001 – Registry Run Keys

Long-term persistence mechanisms

Privilege Escalation

- T1068 – Exploitation for Privilege Escalation
- T1078.002 – Domain Accounts

Targeted domain administrator accounts

Defense Evasion

- T1027 – Obfuscated Files
- T1070.004 – File Deletion

Advanced obfuscation, anti-forensics

Credential Access

- T1003.001 – LSASS Memory
- T1003.006 – DCSync

Mimikatz, DCSync for credential harvesting

Discovery

- T1082 – System Information Discovery
- T1087 – Account Discovery

Comprehensive network reconnaissance

Lateral Movement

- T1021.001 – RDP
- T1550.002 – Pass the Hash

Lateral movement across 250+ systems

Collection

- T1114.002 – Remote Email Collection
- T1005 – Data from Local System

Email theft via OAuth token abuse

Command and Control

- T1071.001 – Web Protocols
- T1573.001 – Symmetric Cryptography

HTTPS C2 with domain fronting

Exfiltration

- T1041 – C2 Channel
- T1048.003 – Unencrypted Protocol

40GB+ data exfiltration

Impact

- T1485 – Data Destruction

Minimal – focused on espionage

Detection and Response Framework

Implementing ATT&CK-Based Detection

1. Coverage Mapping

Create a matrix mapping your security controls to ATT&CK techniques:

Example Coverage Matrix:

Technique	EDR	SIEM	FW	IDS	DLP	Coverage %
T1566 (Phishing)	✓	✓	✓	✓	✓	100%
T1003 (Cred Dump)	✓	✓	✗	✓	✗	75%
T1071 (Web Proto)	✓	✓	✓	✓	✗	80%
T1486 (Ransomware)	✓	✓	✗	✓	✓	85%

2. Detection Analytics Development

Develop detection rules mapped to ATT&CK techniques:

Example: Detecting Credential Dumping (T1003.001)

name: LSASS Memory Access Detection

technique: T1003.001

data_sources:

- Process monitoring

- Windows Security Event Logs

detectionlogic: |

  SELECT * FROM processaccess

  WHERE targetprocess = "lsass.exe"

  AND sourceprocess NOT IN (known goodlist)

  AND accessmask CONTAINS "0x1410"

severity: CRITICAL

false positiverate: Low

responseaction:

- Isolate system

- Dump memory

- Investigate

3. SIEM Use Cases

Use Case | ATT&CK Technique | Data Sources | Priority

- Multiple Failed Logins – T1110 – Event ID 4625 – High
- Suspicious PowerShell – T1059.001 – Sysmon Event ID 1 – Critical
- Unusual Network Connections – T1071 – Netflow, Firewall logs – High
- Mass File Encryption – T1486 – File access logs, EDR – Critical
- Credential Dumping – T1003 – Sysmon Event ID 10 – Critical

Incident Response Using ATT&CK

Phase 1: Detection & Analysis

```
Identify initial alert/indicator

Map to ATT&CK technique

Identify associated tactics

Search for related techniques in same tactic

Build attack timeline
```

Phase 2: Containment

```
Persistence: Remove persistence mechanisms

C2: Block C2 infrastructure

Lateral Movement: Isolate affected systems
```

```
Exfiltration: Block outbound connections
```

Phase 3: Eradication & Recovery

```
T1003: Reset all credentials, enable Credential Guard

T1486: Restore from backups, patch vulnerabilities

T1071: Update firewall rules, block malicious domains
```

Mitigation Best Practices

Layered Defense Strategy

1. Prevention Controls (Pre-Breach)

Control Layer | Technologies | ATT&CK Tactics Addressed

- Network Perimeter – Firewalls, IPS, Web Filters – Initial Access, C2, Exfiltration
- Email Security – Anti-phishing, Sandboxing, DMARC – Initial Access
- Endpoint Protection – Next-gen AV, Application Control – Execution, Persistence
- Identity & Access – MFA, PAM, Zero Trust – Initial Access, Credential Access
- Vulnerability Management – Patching, Virtual Patching – Privilege Escalation

2. Detection Controls (During-Breach)

- EDR – Execution, Persistence, Defense Evasion
- NDR – C2, Exfiltration, Lateral Movement
- SIEM – All Tactics
- UEBA – Credential Access, Collection
- Deception – Discovery, Lateral Movement

3. Response Controls (Post-Breach)

- SOAR – Automated response
- Forensics – Investigation
- Threat Intel – Attribution, Prevention
- Backup & Recovery – Business Continuity

Security Control Effectiveness

High-Value Mitigations

Multi-Factor Authentication (MFA)
Blocks: Initial Access, Credential Access, Lateral Movement
Effectiveness: 99.9% reduction in account compromise
Priority: Critical

Endpoint Detection and Response (EDR)
Effectiveness: 85% detection rate
Priority: Critical

Network Segmentation
Effectiveness: 70% reduction in blast radius
Priority: High

Application Whitelisting
Effectiveness: 90% reduction in malware execution
Priority: High

Privileged Access Management (PAM)
Effectiveness: 80% reduction in privilege abuse
Priority: Critical

ATT&CK-Informed Security Roadmap

Year 1: Foundation

- Comprehensive logging
- Deploy EDR
- SIEM establishment
- MFA implementation
- ATT&CK mapping

Year 2: Enhancement

- NDR deployment
- SOAR automation
- Purple team exercises
- Expanded SIEM use cases

Year 3: Optimization

- **95% ATT&CK coverage**
- **Threat hunting**
- **UEBA**
- **Red team assessments**

Conclusion

The MITRE ATT&CK Framework has revolutionized cybersecurity by providing a common language and structured methodology for understanding adversary behavior.

Key Takeaways

1. Framework Adoption is Essential
2. Defense-in-Depth Remains Critical
3. Detection and Response are Paramount
4. Continuous Improvement is Necessary

Final Thoughts

The MITRE ATT&CK Framework is not just a knowledge base—it's a methodology for understanding, detecting, and defending against cyber threats.

References and Resources

Official MITRE Resources

- MITRE ATT&CK Website https://attack.mitre.org/
- ATT&CK Navigator https://mitre-attack.github.io/attack-navigator/
- ATT&CK for Enterprise https://attack.mitre.org/matrices/enterprise/
- MITRE D3FEND https://d3fend.mitre.org/
- ATT&CK Evaluations https://attackevals.mitre-engenuity.org/

ATT&CK Training and Certifications

- MITRE ATT&CK Defender™ (MAD)
- SANS SEC599
- Offensive Security (OSCP)

Tools and Frameworks

- Sigma
- Atomic Red Team
- Caldera
- VECTR

Community Resources

- ATT&CK Community
- Twitter: @MITREattack
- GitHub: mitre-attack

Research Papers and Reports

- Picus Red Report 2025
  https://www.picussecurity.com/resource/report/red-report-2025

- IBM Security X-Force Index

  https://www.ibm.com/thought-leadership/institute-business-value/en-us/report/2025-threat-intelligence-index

- Verizon DBIR
  https://www.verizon.com/business/resources/reports/dbir/

- CrowdStrike Global Threat Report
  https://www.crowdstrike.com/en-us/global-threat-report

- Mandiant
  M-Trendshttps://services.google.com/fh/files/misc/m-trends-2025-en.pdf

Acknowledgments

This guide synthesizes research from:

- MITRE Corporation
- Global cybersecurity community
- Real-world incident response findings
- APT group tracking research