APPENDIX

Attack Tools

Attackers use a wide variety of tools to accomplish their objectives. This appendix lists many classes of tools used, as defined by function.

Although it is useful to make clear-cut distinctions for classification purposes, tools may serve multiple purposes and span categories.

Antivirus Defeats

Antivirus defeats are technologies or techniques designed specifically to circumvent antivirus heuristic detection. Some are passive, such as not performing a flagged behavior like opening a network connection. Some rely on "getting Admin" or elevating privileges and then performing actions that are ignored in a privileged context. Still others actively attack the antivirus program itself.

Antivirus defeats are required during the initial access, persistence, and expansion phases of an operation. They are essential to maintaining operational and program security.

Network Attacks & Exploitation: A Framework. Matthew Monte

Audio/Webcam Recording

Audio and video recording capabilities may be leveraged during the collection and exfiltration phase of an operation. Though it makes headlines, most widespread malware does not bother collecting this type of data. It's usually too much data to exfiltrate and too much to analyze for too little value.

Backdoor

A *backdoor* or *implant* is a piece of software, hardware, or modification to an existing piece of software or hardware that enables the Attacker to circumvent security. Gaining initial access can be a difficult process that may require using ephemeral or temperamental vulnerabilities or relying on a gullible target. A backdoor ensures future access without the hassle.

Backdoors are an essential form of precaution. Their specific features are fundamental to operational security. They come in two main forms: interactive and noninteractive.

Interactive Backdoor

An *interactive backdoor*, enables the Attacker to execute commands, push or pull files, capture screen shots, or perform other actions in near real time. This requires one or more active network connections between the target box and the Attacker's infrastructure. It also requires a human Attacker on the other end of the connection to send instructions.

An interactive backdoor is usually required to expand access, survey, or make any real-time operational decision.

The Poison Ivy RAT (Remote Access Tool) is an example of an interactive backdoor.

Noninteractive Backdoor

A *noninteractive backdoor* is also known as *beaconing malware*. The undesirable program calls out and establishes a connection to a website, mailbox, file drop, chat channel, or other virtual location. It then retrieves and executes any tasking placed there by the Attacker. A beaconing backdoor may also be used for exfiltrating data. Collected data is leaked out over time to one or more locations.

This form of command and control is required to establish a connection with a target behind a firewall, NAT, or other form of network security. It is necessary for persistence within a network.

Noninteractive command and control provides the Attacker with the ability to scale, as a person is not required to drive each implant. It is the most prevalent form of malware.

The Zeus Trojan is an example of a noninteractive backdoor.

Bootkit

A *bootkit* is a specialized backdoor that loads during the computer boot process. Bootkits often can circumvent core operating system security and restrictions because they are loaded before the operating system in the boot process.

Bootkits are a form of deep precaution, as some may even survive the complete reset of a device and fresh reloading of the operating system. Bootkits also help maintain operational security.

Collection Tools

A *collection tool* is a catchall category for any tool that gathers information for exfiltration to the Attacker. A collection tool may collect e-mail, browser history, keystrokes, passwords, images, engineering drawings, word documents, or anything that is stored on or transits a computer. As the name implies, collection tools fulfill the mission objectives for strategic and directed collection.

Exploits

Exploits are pieces of software that leverage a vulnerability in software or hardware to perform a restricted action. Exploits fall into the three basic categories described in the following sections.

Remote Execution

As the name implies, a *remote execution exploit* enables the Attacker to execute code remotely on the target machine. The exploit may be Attacker initiated whereby the Attacker connects into the target machine. These are known as "pure" remote exploits. Exploits may also be target initiated in which the target is required to connect to an Attacker-controlled service. These are known as "client-side" exploits.

Client-side exploits are also known as *malicious server exploits* because they require the Attacker to set up or compromise a server that the target will access. A specialized subclass of malicious server exploits is cross-site scripting vulnerabilities. In these, the Attacker implants code on a third-party site for distribution to potential targets without compromising the security of the site itself.

Remote execution exploits are required for gaining initial access and sometimes for expanding access within a network.

Local Privilege Escalation

A *local privilege escalation* grants the Attacker elevated access to a resource. A common example is escalating from a standard user on Windows to the Administrator or from the Local Administrator to the Domain Administrator.

Local privilege escalations are required to gain the kind of access the Attacker needs to circumvent antivirus or other defensive products, expand access through a network, or install persistently on a computer or device.

Information Disclosure

In an *information disclosure exploit*, the Attacker gains no control over the target computer but rather retrieves what is supposed to be restricted information.

A common information disclosure abuses a poorly configured or insecurely coded website that accesses a back-end database. Through use of weak or poorly secured credentials or a technique called SQL injection, the Attacker can bypass any front-end security and access the database. For example, Andrew Auernheimer, aka "weev," exploited a rather trivial information disclosure vulnerability in an AT&T website that allowed him to retrieve 100,000 iPad users' e-mail addresses.

Information disclosures are used during targeting and initial access. Depending on the operational objective, they may also be the ultimate goal of the operation.

Fuzzer

A *fuzzer* is an automated or semiautomated program for finding vulnerabilities. Fuzzers input randomized data into targeted software or hardware and then check for indications of flaws such as memory leaks and program crashes. A fuzzer's success depends on the quality of the technology tested and how well the fuzzer stresses it.

Fuzzers are a means to economize resources and to increase knowledge. In the worst case, they might reduce the tediousness and resource drain of testing to find simple vulnerabilities. In the best case, they could uncover vulnerabilities that would otherwise not be found.

Hardware-based Trojan

A *hardware-based Trojan* is when a backdoor is implanted directly into the circuitry of the target equipment. This type of tool is what you might think of in an old spy movie where a phone is "bugged." The listeners slipped in during the middle of the night to make some modification to the physical phone to record audio. That sounds almost quaint by today's standards.

The real threat is the Chinese implanting computer chips with backdoors during the manufacturing process,[1] the NSA fashioning malicious USB cables,[2]

or some independent researcher making a USB charger that steals wireless keystrokes.[3]

Such tools are masters of operational security. Hardware is simply outside the scope of most defensive security technology. The only saving defensive grace is that hardware Trojans require the Attacker to gain physical access to the target or their supply chain.

Implant

See "Backdoor" earlier in this chapter.

Keystroke Logger

A *keystroke logger*, or *keylogger* for short, captures and records keystrokes as the user types them. Keystroke loggers are one of the most basic offensive tools. They may be general, capturing all key presses, or application specific, capturing only those within a specific program such as a web browser.

Keyloggers are most useful in capturing logins, passwords, and any other information that is entered but not stored on the computer. This information is not only essential for expanding access, but also for maintaining precaution. Keeping an up-to-date list of passwords helps guarantee access.

This tool may also be used for collection, such as grabbing encrypted messages before they are encrypted, or e-mails that are written but never sent or saved.

Network Capture

A *network capture* tool records network traffic to and from a target machine. This is useful during expansion as the Attacker may pull passwords or password hashes out of the connection stream. Network capture tools may also be useful for passively mapping the network by finding e-mail servers, web proxies, and more.

Network Survey

A *network survey* tool maps out both static and dynamic views of a target network. The static map can show computers, their operating systems, configurations, and offered services, as well as networking gear such as routers, firewalls, and Wi-Fi access points. A static map also shows how these various devices are interconnected, the hierarchical organization, the network addresses, subnets, and more. The dynamic part of a network survey includes gathering information on how data is routed into, out of, and through the network; the bandwidth; any mobile or intermittent network clients; the server load; and so on.

Often a network survey requires a variety of tools tailored to profiling specific types of network setups and devices. Maintaining an accurate survey of the network is an ongoing task and fundamental to achieving any level of target awareness.

Network Tunnel

A *network tunneling* tool is required to route data between the Attacker and different segments of a target network. For example, as shown in Figure A.1, many corporate networks have several tiers: a DMZ for Internet-facing servers, a user network, an internal server network, and so on.

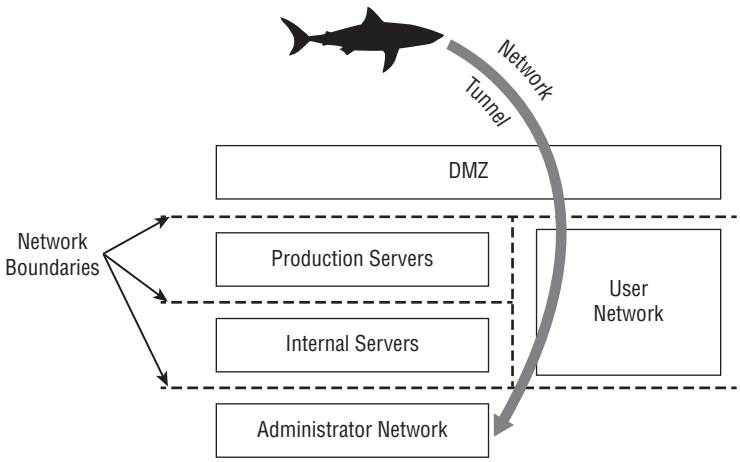

Figure A.1: Network tunneling

Attacker points of access can be anywhere but usually begin either in the DMZ or on the user network. A network tunnel enables Attackers to route through their points of access to an internal end point to expand access or collect data.

Password Dumpers and Crackers

A *password dumper* is a utility that pulls clear text or hashed passwords from a target system. Password dumpers exist for practically every platform, including desktop computers, mobile phones, and a variety of networking equipment.

Hashed passwords are passwords that are run through a one-way mathematical algorithm before being stored. In general, a hashed password is not particularly useful, though there are exceptions. To be useful, the Attacker must recover the original password from the hash.

If the hash is implemented correctly, reversing the hash and recovering the original password is mathematically impossible. Instead, Attackers employ *password crackers*. A password cracker uses a combination of storage, processing power, and intelligent search algorithms to attempt to brute force or guess the

original password. For each potential guess, the cracker computes a hash and compares it to the dumped hash. If they match, the Attacker knows the guessed password matches the original.

Password dumpers and crackers are essential to expanding access. A common Attacker tactic is to gain access to an administrator's machine, dump his passwords, and then use the administrator's credentials to move throughout the network.

Packer

A *packer* takes a piece of software; bundles it into a payload; and compresses, encrypts, or otherwise obfuscates that payload. When run, a packed program reverses the process and decompresses, decrypts, and then loads and runs the payload.

Application installers are a simple nonmalicious form of a packer. When the user double-clicks a typical application installer, it decompresses and copies embedded files to the appropriate folders, and then makes any required system or settings changes.

Malicous packers are conceptually no different in function than an application installer, except their purpose is to avoid analysis. Simple packers decrypt everything at once. Advanced packers decrypt portions as needed, leaving only a small portion of the payload visible at any given moment.

Packers provide a way to improve operational security by avoiding antivirus signatures or network filters. They also may improve program security by increasing the cost of analysis, slowing down analysis, and limiting the amount of useful information that can be disseminated.

They do have their downsides, though. The portion of the packer responsible for packing or unpacking the payload may be signatured, leading to a decrease in operational security and a linking of different tools. To counter this defect, Attackers have increasingly resorted to one-off packing utilities or polymorphic code generators.

Persistence Mechanism

A *persistence mechanism* is the method a backdoor, collection tool, or other program uses to start running. It is not a tool, per se, but rather the functionality required to make other tools work. A backdoor is useless if it is never run.

Persistence mechanisms are not inherently malicious. Many are legitimate and thoroughly documented for use by developers. A persistence mechanism may be used by legitimate programs to, for example, provide hardware support, run antivirus scans, check for software updates, perform backups, or more.

Persistence mechanisms operate at several different privilege levels and are run during different parts of the startup process, as shown in Figure A.2. At the lowest level are hardware-based implants. These hook directly into the physical device's startup process.

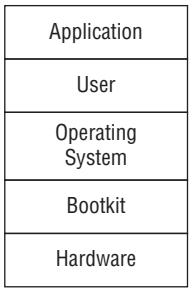

Figure A.2: Persistence levels

The next level up is a *bootkit*, which hooks into the software routines used to load an operating system. The TDL or Alureon rootkit is an example that hijacks a Windows computer's Master Boot Record to load malicious code before the operating system.

Further up the stack are programs that launch when the operating system is booted. There are a wide variety of these on every operating system. On desktop systems, these include drivers, system services, and scheduled tasks. The Windows Firewall is a program launched by an operating system–level persistence mechanism. According to Mandiant, the so-called BISCUIT malware used by APT1 is also a service.

Next, there are user-level persistence mechanisms. These launch when a user logs into the system. The Startup folder on Windows is an example, as is the Login Items setting on Mac. Many versions of malware use user-level persistence mechanisms as a fallback when they do not have sufficient privilege to install at a lower level.

At the highest level are application-specific persistence mechanisms. These are launched with or as part of the specific application. A common legitimate example is a browser plugin, such as Java, that launches when the user visits a certain page. A quasi malicious example are the browser toolbars that many shareware programs install.

The different levels of persistence mechanisms have trade-offs between ease of implementation, potential functionality, and stealth. At the bootkit level, a program could reprogram the operating system to capture user logins, but it will be hard pressed to capture a screen shot. Graphics are running on a completely different level of the technical stack. A hook into user level can easily grab the screen shot, but it will have limited options for stealth. User-level programs do not have many of the privileges required to avoid detection.

Persistence mechanisms are one of the few natural choke points in the Attacker's toolkit. The Attacker needs something to persist to maintain access, even if it only operates intermittently. For this reason, security programs attempt to monitor, detect, and prevent the installation of anything persistent.

A wide variety of persistence mechanisms are essential for maintaining precaution, operational security, and program security.

Polymorphic Code Generator

A *polymorphic code generator* takes computer software in source code or machine code format and transforms it so that the code is changed but the underlying functionality remains the same.

For a trivial example, suppose a program adds three numbers together. Sample code might look like the following:

```
sum = a + b + c
```

This is one way of doing it, but an equivalent action might be

```
sum = b + c + a
```

or even

```
intermediate = (a - 42 + c) * 2
does_not_matter = intermediate * 5
save(does_not_matter)
sum = intermediate/2 + b + 42
…
delete(does_not_matter)
```

The latter set of instructions is inefficient, but it serves the purpose of altering what the generated binary code will be. Different binary code means a different look to static antivirus scanning.

A polymorphic code generator automates a morphing process like this example for entire programs, generating different output programs with the same malicious functionality as the input program.

As detailed by Brian Krebs, some enterprising criminals have even created a moneymaking service out of combining polymorphic code generators, packers, and offline antivirus scanning.

> *[A] crypting service takes a bad guy's piece of malware and scans it against all of the available antivirus tools on the market today—to see how many of them detect the code as malicious. The service then runs some custom encryption routines to obfuscate the malware so that it hardly resembles the piece of code that was detected as bad by most of the tools out there. And it repeats this scanning and crypting process in an iterative fashion until the malware is found to be completely undetectable by all of the antivirus tools on the market.*[4]
>
> **—Brian Krebs**

It's one-stop shopping for all your criminal antivirus circumvention needs.

Code morphing enables the Attacker to maintain some level of program security at reduced cost. With this capability, the capture of one offensive tool may not lead to the compromise of another, even if the two have identical functionality.

Rootkit

A *rootkit* is a program that hides files, network connections, processes, and more. Rootkits provide the stealth necessary to avoid detection and maintain operational security. Though a rootkit may be paired with any offensive tool, it is almost always paired with a backdoor.

Screen Scraper

A *screen scraper*, or *screenshot tool*, is used to capture the image that appears on a user's screen. Screenshots are useful for being aware of what a user or program is doing at a given moment. This functionality has been part of virtually every backdoor for the past 15 years. Screen scrapers are trivial to implement because Windows and Mac have built-in functions for capturing screen output.

System Survey

A *system survey utility* gathers information about one or more computer systems where the Attacker has access. Survey utilities gather a wide variety of information including files, processes, Registry entries (Windows), network shares, installed programs, detailed operating system information, user lists, uptime (the length of time since a system was rebooted), and more.

Survey functionality is often built directly into backdoors or integrated via modules; though it may be a separate utility as well. Survey functionality is critical to target awareness.

Vulnerability Scanner

A *vulnerability scanner* is exactly that, a tool that scans a network for known vulnerabilities. Scanners are dual-use tools, as the Attacker can use them to find vulnerabilities in a potential target while the Defender can use them to locate issues within their own network.

Vulnerability scanners may be combined with an automated exploitation system that instantly exploits any issues found. Immunity Security's Canvas software and Rapid7's Metasploit provide this kind of integrated functionality.

Vulnerability scanners are useful during the targeting and expansion phases of an operation. They may also double as a form of network survey tool, providing target awareness.