CS5431 – Computer Security Practicum Spring 2017

Lecture 3: Threat Models

February 10, 2017 *Instructor: Eleanor Birrell*

1 High-level Threat Models

A secure system should exhibit all the functionality that the system designer expects and no functionality that the designer doesn't expect. Thus whether a system is secure depends on how principals—in particular how adversarial principals—interact with the system. This set of expected adversarial interactions is defined as the *threat model*. Threat models are characterized of the means, motivation, and opportunity of the attacker.

The DoD Defense Science Board defines a hierarchy of threats ranging from so-called "script kiddies" to sophisticated nation states capable of introducing backdoor vulnerabilities into commercial products and combining sophisticated cyber attacks with military and/or intelligence activities:

- Tier I: Practitioners who rely on others to develop the malicious code, delivery mechanisms, and execution strategy (use known exploits).
- Tier II: Practitioners with a greater depth of experience, with the ability to develop their own tools (from publicly known vulnerabilities).
- Tier III: Practitioners who focus on the discovery and use of unknown malicious code, are adept at installing user and kernel mode root kits, frequently use data mining tools, target corporate executives and key users (government and industry) for the purpose of stealing personal and corporate data with the expressed purpose of selling the information to other criminal elements.
- Tier IV: Criminal or state actors who are organized, highly technical, proficient, well funded professionals working in teams to discover new vulnerabilities and develop exploits.
- Tier V: State actors who create vulnerabilities through an active program to influence commercial products and services during design, development or manufacturing, or with the ability to impact products while in the supply chain to enable exploitation of networks and systems of interest.
- Tier IV: States with the ability to successfully execute full spectrum (cyber capabilities in combination with all of their military and intelligence capabilities) operations to achieve a specific outcome in political, military, economic, etc. domains and apply at scale.

Threat models implicit make assumptions about what an attacker will or will not do; a system is not secured against threats outside of the envisioned attack model because it is assumed that an attacker targeting that system will lack the means, motive, and/or opportunity to exploit such attacks.

2 Capability-driven Threat Models

For the purposes of applied security, it is often useful to take a more concrete approach to threat models. Instead of describing the threat model in vague terms of their DoD-specified Tier, we will focus on the *capabilities* of the attacker. Today, we will consider several different capabilities that (assumed) attackers targeting your system might, or might not, have.

2.1 Privilege Levels

An adversary who successfully executes a code injection or a code hijacking attack—such as the buffer overflow attacks discussed in the last two weeks—can cause the target system to execute code that was not intended by the system designer. The exploit code runs with the same privileges as the target code.

One standard threat model is therefore the class of adversaries who can run exploit code with user privileges but not with root privileges. This models the case in which attackers can successfully target programs that runs as an unprivileged user. Under this threat model, OS-level access control mechanisms can be used to main the security of valuable assets. This approach is commonly used to protect assets like password files an audit logs against this class of adversaries. For example, if the target program is only given read access to a password file, then an attacker who injects or hijacks that program can execute exploit code that reads the password file but will be unable to modify the password file. This prevents this adversary from deleting password entries (causing denial-of-service to the targeted user(s)) or replacing password entries with passwords of their own choosing (enabling access to the targeted account(s)). As long as the harms from reading a password file are mitigated (e.g., by following the standard practice of storing hashed and salted passwords), these techniques comprise successful countermeasures against this class of adversaries. Since this defensive pattern is so common, threat models often specifically consider a class of adversaries who can obtain read access to security-critical file but who cannot write to those files.

Of course, an attacker can bypass such protection mechanisms if they are able to run exploit code as a privileged user. This can be achieved either by targeting a vulnerability in a program that runs as a privileged user (e.g., root) or by means of a *privilege escalation attack.* Privilege escalation works by first gaining access to the system and then acquiring additional privileges by acquiring credentials for a privileged user (e.g., by accessing the password file and cracking an administrator password) or by targeting a process running

with higher privileges (e.g., by executing a code injection or code hijacking attack on such a process).

Dirty COW. Dirty COW is a privilege escalation vulnerability discovered in Linux in October 2016. A bug in the copy-on-write (COW) mechanism allowed any installed application to gain root level access. At a high level, the exploit works by opening a root-owned executable as read-only and memory mapping it into the user-level process's address space. The user-level process then runs two threads: one of which repeatedly uses `madvise()` to tell the kernel it doesn't intend to use that memory and one of which repeatedly overwrites some of its own memory that's mapped to the root-owned executable. A race condition bug results in writes updating the mapping copy instead of writing to a private copy; these changes are subsequently committed by the kernel to storage. This technique can be exploited to alter a root-owned setuid binary so that it, for example, spawns a root-owned shell, allowing the user-level process to operate with root privileges.

2.2 Memory Access

An attacker who gains access to the system will inherently be able to access the disk. Although that access might be limited by the OS-enforced access-control permissions discussed above, the frequency of privilege escalation vulnerabilities causes many high-value systems to consider a threat model in which attackers have full read/write access to disk.

Access to disk does not, however, imply that an attacker has access to memory. Operating systems enforce isolation between the virtual memory spaces of different processes. And even though an attacker could gain some level of access to the memory space of the target application (e.g., using a vulnerable buffer), this is generally limited in practice.

If the threat model includes attackers with access to disk but not access to memory, then sensitive or valuable information (including logs and password files) can be protected by encrypting the values stored on disk. The encryption key cannot be stored on disk—because the adversary is assumed to have full disk access—but can instead be entered by an administrator (or derived from an administrator password) when the system is started. It is, of course, necessary to ensure that sensitive memory is compromised by paging it out to disk; this can be done either by disabling or encrypting swap spaces.

Of course, an adversary outside this threat model—one who can access memory—might be able to bypass these defenses. There are multiple ways in which an adversary might obtain access to memory, including executing a buffer overread attack or using direct memory access (DMA). An adversary with only disk access might also be able to bypass these defenses if memory pages containing keys or passwords are swapped out to disk (without first being encrypted).

Heartbleed Heartbleed was a vulnerability in the OpenSSL resulting from a missing bounds check (a buffer overread vulnerability) in its implementation of the TLS heartbeat extension; it was discovered in April 2014.

Transport Layer Security (TLS) is the standard protocol for creating secure, encrypted communication channels. The TLS heartbeat extension is designed to keep the connection alive, avoiding the overhead of negotiating a new connection; it works by having one end of the connection send an arbitrary message which the other end echoes back, proving that the connection is OK. The heartbeat message, according to the RFC, has the following structure:

```
struct {
    HeartbeatMessageType type;
    uint16 payload_length;
    opaque payload[HeartbeatMessage.payload_length];
    opaque padding[padding_length];
} HeartbeatMessage;
```

Unfortunately, OpenSSL's implementation did not check the claimed `payload_length` against the actual length of the TLS record sent. If the claimed length (which could be up to 64KB) exceeded the length of the actual payload (which could be just one byte), the "echo" message returned by the server would send back a copy of the one byte message as well as the next 64KB-1 bytes of memory. Because this memory often contains sensitive values, this approach allowed an attacker to access servers' private keys as well as users' session cookies and passwords. At the time it was disclosed, 17% of secure web servers on the Internet were believed to be vulnerable. Experts estimate that as of January 2017, as many as 200,000 services remain vulnerable.

2.3 Physical Access

Another common distinction to make in terms of adversary capabilities is the distinction between adversaries who have only remote (i.e., network) access to a machine versus those who have physical access to a machine.

For high-security systems, if the attacker is assumed to not have physical access to the machine, then "air gapping" a machine is an effective defense. An airgap is achieved by physically isolating a machine from all untrusted networks. If the attacker's only way to access the target system is by connecting from an untrusted network, then an air gap is a 100% successful defense.

Stuxnet Stuxnet was a computer worm first identified in 2010. It initially spread using infected USB drives to bypass air gaps, and it targeted programmable logic controllers (PLCs) used to control machinery, specifically Iranian centrifuges. Widely believed to have been a join effort by American and Israeli intelligence organizations, Stuxnet was notable both for its use of four separate zero-day vulnerabilities and for the physical

damage it caused by forcing changes to centrifuges' rotor speed.

Physical access can also be used to acquire additional capabilities. For example, an attacker with physical access to a machine might be able to leverage that access to acquire access to memory or to encryption keys using a cold boot or side channel attack.

2.4 Key Access

A common assumption is that there are some secrets (usually encryption keys and/or passwords) that the attacker cannot gain access to. As long as these secrets remain secure, the system remains secure; attackers who can compromise those keys are outside the system threat mode.

Encryption keys can be used in a variety of ways including to maintain the confidentiality and integrity of disk files, to maintain the confidentiality and integrity of network communications, and to maintain the confidentiality and integrity of audit logs.

If keys are stored at a third-party provider and the adversary has the power to subpoena that information, then the adversary can access those encryption keys. This is the case, for example, if a user enables full-disk encryption using Apple's FileVault but elects to store a recovery key in their iCloud account. In the United States, this capability to issue subpoenas can generally be bypasses if the key (or, in the case of password-based encryption, the password used to derive the keys) is known only to the data owner, since passwords have generally been viewed as protected under the First and/or Fifth Amendments.[1]

There are also technical means to obtain encryption keys. Some of these approaches depend on having access to memory and/or physical access to the computer. If keys are derived from passwords, they can again be acquired through memory access or physical access or by tricking the user with phishing or social engineering attacks.

Bypassing FileVault 2 FileVault 2 is Apple's current full-disk encryption software. It works by encrypting the disk using XTS-AES under a 256-bit key; the key can be recovered by using the (a) user account password as a passphrase. Since the disk is always encrypted, attackers cannot learn sensitive information by reading the disk directly; plaintext data is only ever stored in memory. Direct memory access (DMA)—a feature that allows certain hardware to access memory independent of the CPU—is prevented using Intel's Virtualization Technology for Directed I/O (VT-d), which can restrict device DMA to particular regions of physical memory.

However, in December 2016 it was observed that the security offered by FileVault could be bypassed due to the combination of two different bugs. First, in the early stages of the boot process (before the OS was started), the defenses against DMA were inactive. As a result, hardware connected to a Thunderbolt port (which enables DMA

[1] A December 2016 ruling by the Florida Court Appeals made this legal argument more ambiguous; final determination is likely to rest with the Supreme Court.

over PCI express) could directly access main memory. Second, the user password (entered in order to access the machine) was never deleted from memory, so when the computer was rebooted, the password was still accessible in memory. As a result, an adversary who was capable of physically accessing a machine that was not shut down (including a machine that was logged out or in sleep mode) could retrieve the user password and access the full, decrypted system, thereby accessing information that was only available in memory.

The iPhone Case In December 2015, a terrorist attack killed 14 people in San Bernardino, California. The FBI recovered an iPhone belonging to one of the shooters, but it was encrypted and could not be unlocked without entering the PIN. If too many incorrect attempts were made to access the phone, the data would be wiped. The federal government tried to compel Apple to write and sign new software that would accept bypass the need for the correct PIN, but Apple challenged the order in court. The case was dismissed after the FBI acquired alternate means of accessing the data on the phone.

FInally, keys or passwords can be acquired by tricking a user into sharing their credentials using a phishing, spear phishing, or social engineering attack.

U.S. Election Hacks In March 2016, John Podesta, the chairman of Hillary Clinton's presidential campaign, received an email claiming to be from Google stating that someone had attempted to access his gmail account from an IP address in Ukraine and that he should change his password. After checking with the campaign help desk, he clicked on the link and proceeded to enter his old password and his new password, unknowingly handing his credentials to the attacker. This targeted phishing attack—believed to be the work of Russian intelligence—resulted in the publication of thousands of emails on WikiLeaks in October 2016. The compromised emails, which included controversial excerpts from Clinton's Wall Street speeches, are considered by some to have contributed to her loss in the presidential race.

2.5 Network Access

In addition to remote or physical access to the system and/or the computer(s) on which the system is running, an attacker is likely to have some level of network access. This might include the ability to read packets, generate new packets, and/or intercept packets in transit.

One standard threat model assumes that attackers are capable of reading, writing, and intercepting any packet at any time; this is sometimes called the Dolev-Yao threat model. It is commonly adopted when designing cryptographic models because it is seen as a "worst-case" model. That is, if you can prove a cryptographic protocol is secure even against a Dolev-Yao model then it will surely be secure against any real-world attacker.

Whether the Dolev-Yao threat model is appropriate for a particular system depends again on the network capabilities of the adversaries against whom the system should remain secure. An adversary can read packets if it has access to the local area network (LAN) of one of the endpoints; with physical proximity, read access can be acquired by connecting to a wireless router in promiscuous mode, by physically connecting to the network, or by spoofing the target's MAC address. An adversary can also read packets if it has control of one or more of the routers in the network; this is generally not feasible for individuals, but it is a reasonable capability for a nation state.

Any adversary can create new packets. How many packets an adversary can generate depends on the resources available to the attacker; an adversary who can generate enough packets can overload a target system resulting in a Denial of Service (DoS) attack that compromises the availability of the system.

An adversary can intercept packets if it controls routers in the network or if it can spoof the target's identity on a local network. An adversary can also cause uncompromised routers to drop network packets by overloading them with packets.

Pakistani YouTube In February 2008, Pakistan decided to block access to YouTube due to the increasing number of videos it considered blasphemous. The Pakistani ISP Pakistan Telecom implemented this ban by routing the address block 208.65.153.0/24 to a black hole. Due to the disambiguation protocol in the Classless Inter-domain Routing (CIDR) protocol, this routing information was propagated to Pakistan Telecom's parent ISP in Hong Kong, which propagated it to the rest of the world, rendering YouTube unreachable to most of the Internet.

DDOSing Dyn In October 2016, a network of about a hundred thousand internet-connected devices using millions of IP addresses flooded Dyn.com with requests. The devices had been compromised by Mirai, a piece of malware that spreads by logging into devices over telnet or SSH using the default password; there are an estimated 1.2 million Mirai-infected devices on the Internet today. The coordinated, distributed denial of service (DDoS) attack took Dyn's nameservers offline. Since Dyn provides DNS services for many companies, this resulted in outages in many companies including GitHub, Twitter, Reddit, Netflix, and AirBnb.