Descriptions of commonly used terms in cyberattacks & vulnerabilities

This document gives an overview of terms which are commonly used in lectures related to cybercrime and attacks. It is intended as a summary of selected topics from the courses 2IMS20 (Cyberattacks, crime and defenses) and 2DMI20 (Software security).

Common vulnerabilities and attack types

GET and POST parameters/requests

HTTP GET requests are intended for requests that *do* ***not*** modify state. POST requests are intended for requests which *do* modify state. GET requests have their parameters visible in the URL; this means that these parameters are stored in browser history, and could also be part of a link (e.g. one sent via a phishing email). POST request parameters are not visible in the URL of a page, and hence do not get stored in browser history and cannot be submitted as part of a link. Instead, they are submitted via forms on a webpage. Still, it must be noted that forms can also be submitted via another site, which means that, regardless of the request type, CSRF attacks are possible.

SQL injection

SQL Injection is a vulnerability which occurs in web applications. Typically, SQL injections are possible because user input is inserted into SQL/database queries via string concatenation. This allows for specifically crafted user inputs to be interpreted as code instead of data. Given that attackers often have the ability to provide user input to the application, this effectively allows attackers to read, manipulate and/or destroy arbitrary data in the database.

There are several ways to prevent SQL injection. The most obvious one is to access the database using so-called *prepared statements*. In such statements, the part which is intended to be interpreted as code is pre-compiled so that it (and only it) is interpreted as code; the user inputs are filled in as parameters which can only be interpreted as data. Other ways to prevent SQL injection include filtering user input to ensure it does not contain any characters that could cause (parts of) the input to be interpreted as code or storing procedures at the database server.

XSS

XSS, or cross-site scripting, is an attack in which attacker-provided JavaScript code is executed on the victim's computer, within the context of a site not (intended to be) under the control of the attacker. This allows the attacker-provided code to access data such as cookies which should only be accessible to the site which stored that data.

There are two (main) variants of XSS. The first one is stored XSS, in which the attacker-provided JavaScript is stored on the attacked site, and then served whenever the victim opens the page. This variant generally occurs whenever attacker-provided inputs are not properly sanitized by the server before they are stored. A typical example would be a comment form which does not verify whether the submitted comment does not contain HTML/JavaScript code.

The second variant is reflected XSS. In this variant, the attacker-provided code is not stored on the attacked site, but is provided by the victim (often in the form of a link with GET parameters) to the attacked site and then echoed back in the response. It mostly occurs when user inputs are (1) echoed back to the user and (2) when those inputs are not properly sanitized before being echoed back.

The most common countermeasure to XSS attacks is to properly sanitize user inputs before storing them (in the case of stored XSS) or echoing them back (in the case of reflected XSS). The crucial thing is that `<script>` tags need to be removed so that the browser cannot interpret the code as JavaScript.

Usually, the aim of an XSS attack is to read cookies for the attacked site (which can only be done using a JavaScript provided by that site) to then send them to the attacker.

A third type of XSS is server-side XSS, in which the attacker is able to make the server access a website/server on their behalf. This usually does not involve JavaScript, but can still be dangerous, as it allows the attacker to bypass firewalls.

Cookies

SUPERCOOKIES & THIRD-PARTY COOKIES

Supercookies are a term for 'cookies' which cannot be deleted through regular means; in fact, they are not cookies at all. Instead, they are other means through which a website can identify a recurring user. Originally, the term has mainly been used for headers inserted by ISPs which identify a user with a unique ID. However, the term can also be used for e.g. IP address-based tracking or tracking based on HSTS.
Third-party cookies are cookies for third-party resources (i.e. those hosted on a site different to the one being accessed) which are accessed on a webpage. Although this nowadays depends on the browser being used, disabling third-party cookies (at some point) only prevented third parties from storing cookies, and not from accessing cookies already stored. This still allowed for tracking.

Watering hole attack

A watering hole attack is an attack where the intended victim is not directly infected; instead, a site/resource commonly accessed by the intended victim is compromised. Then, whenever the victim accesses this resource, they can be (for example) infected with malware.
The main idea behind this attack is that it may be difficult to directly infect a victim, especially if they are well aware of phishing and other attacks. However, by instead compromising a site the victim is likely to visit, it is possible to find an attack vector much easier.

Double & triple extortion

In ransomware, double extortion refers to the attacker strategy where a ransom is requested for two purposes:
1. Decrypting files, i.e. restoring their availability
2. Not publishing the files, i.e. maintaining their confidentiality.

In triple extortion, not only the compromised organization is asked to pay a ransom; instead, a ransom is also requested from people whose (sensitive) data has been breached. This allows cybercriminals to earn more money from a single attack.

Other relevant concepts (for defenders)

Network priority triangles

FOR IT & ICS NETWORKS

For IT networks, the security properties have importance in the following order (from most important to least important):
1. Integrity
2. Confidentiality
3. Availability

For ICS networks, the order would be:
1. Integrity/Availability (both are equally important)
2. Confidentiality

Cyber kill chain

The cyber kill chain consists of 7 phases:
1. Reconnaissance
2. Weaponization
3. Delivery
4. Exploitation
5. Installation
6. Command & Control (C2)

7. Action on objectives

Living off the land attack

A living of the land attack is an attack where intruders use legitimate software (e.g. Windows Management Instrumentation, PowerShell) to perform malicious actions on the system. Their main advantage lies in them being much more difficult to detect. Their disadvantages include speed and cost.

Living off the land attacks are particularly common to ICS networks; due to the wide variety in software and protocols in use there, it often does not make sense to create tailor-made malware.

Supply chain attacks

In a supply chain attack, a supplier of the target/intended victim is attacked. The aim is to introduce a piece of malware in software from the supplier, which then infects the customer of the supplier (i.e. the target) as soon as the software is updated. Supply chain attacks have 6 advantages:

1. They allow for infecting well-protected networks *(since the malware comes in through a 'trusted' vector)*;
2. They spread quickly and on a vast scale through automated updates;
3. They can be targeted to affect certain regions and/or sectors;
4. They allow for infection of isolated networks *(i.e. networks not connected to the internet)*;
5. They are difficult to detect, since the problems originate from 'trusted' software;
6. They often allow for privileged access to systems, since software update mechanisms are often executed with administrative privileges.

Software Bill of Materials

A software bill of materials includes (possibly in machine-readable format) a list of the software a given piece of software depends on (possibly also indirectly). The aim of a software bill of materials is to make it easier to manage dependencies, and to make it easier to act quickly when security vulnerabilities are discovered in one of them, i.e. by updating the vulnerable software. In particular, a software bill of materials makes it easier to identify vulnerable systems and to act on that knowledge. Finally, software bill of materials is particularly relevant to IoT systems, since they suffer from issues such as vendors going out of business, being publicly exposed and being difficult to patch in general.

Targeted attacks

Stuxnet

Stuxnet was the first piece of malware to target industrial control systems (ICS). It combined several zero-day vulnerabilities to eventually target Iranian nuclear centrifuges. The attack roughly consisted of three phases:

1. Spread as a worm
2. Attack Siemens/PLC system
3. Sabotage

We start with the third, and final phase. In that phase, the malware altered the spinning frequency of centrifuges in nuclear facilities, with the aim to break them. Simultaneously, the malware provided false feedback to other systems (which were read out by operators) in an attempt to hide its existence. *(In effect, Stuxnet was therefore the first rootkit for a PLC.)*
To achieve its goals in the third phase, Stuxnet first had to reach the Siemens PLCs used in the nuclear facility to control the centrifuges. To do so, it first infected software used to manage said PLCs in the second stage.
Before that, however, the malware had to reach a computer which used the PLC software to update the PLCs. For this part, which encompasses the first stage, it behaved like a worm. This worm spread on its own from infected computers, both via the network (e.g. through Windows vulnerabilities in, for instance, the print spooler) and via USB sticks (by using a vulnerability involving LNK (Windows shortcut) files. This latter part was important; since the nuclear facility's network was not linked to the internet, it was necessary to use vulnerabilities in USB sticks (or,

more precisely: the software handling those USB sticks) to get onto the target network in the first place.

One of the unique aspects of Stuxnet is its level of targeting: it only performed its malicious actions in a very specific system (with equipment from certain manufacturers *and* centrifuges operating within a specific frequency range) and left other systems (nearly) intact. This was most likely done to hide the malware and prevent it from being discovered before it could do its job. In addition, this (combined with the PLC rootkit) made it much more difficult to detect the malware in the first place (since equipment failure would be a more likely cause of the issues at hand).

Havex

Havex was a piece of malware which was intended for reconnaissance purposes and to gain persistent access. It most likely originates from Eastern Europe. It entered systems through e.g. spear phishing, watering hole and supply chain attacks. It mostly targeted ICS networks (in particular: energy companies).

BlackEnergy 3

BlackEnergy 3 was a piece of malware used as part of a campaign to cause power outages in Ukraine in late 2015. Its initial infection vector was spear phishing with malicious Word documents. Once it got in, it retrieved VPN credentials to access the network more stealthily (compared to C2). Within the network, malicious firmware was deployed to serial-to-ethernet devices (to make recovery more difficult). In addition, KillDisk was used to wipe servers. These two pieces were activated once the attack commenced: at that point, breakers were open to cause a power outage and communication with customer service was made more difficult by means of DoS against phone services.

Industroyer

Industroyer was a piece of malware used to cause power outages in Ukraine in late 2016. The main differences with BlackEnergy 3 include it being more modular and the malware working in a mostly automated fashion this time.

Industroyer 2

Industroyer 2 is a piece of malware used for similar purposes as Industroyer. Its main difference lies in its behavior being more targeted: its configuration allows it to specify its behavior according to the target environment it has handed up in. This makes it easier for the malware to be applicable in a multitude of environments.s

CryptoLocker

CryptoLocker was the first ransomware which presented a reproducible business case. In other words, it was the first ransomware thought to be able to make a profit. It appeared in 2013. A trick used to masquerade CryptoLocker's executable files in email attachments was to have the files include double extensions (e.g. `document.pdf.exe`). It spread over the Gameover ZeuS botnet.

Petya

Petya is a piece of ransomware. Its main distinguish aspect is that it does not encrypt files; instead, it replaces the master boot record and encrypts the file table of an NTFS volume to make files unfindable.

WannaCry

WannaCry is a piece of ransomware which appeared in 2017. The main aspect making this ransomware different from others was that it was able to spread on its own (i.e. it is a worm). For this, the EternalBlue vulnerability, which was allegedly developed by the NSA, was used.
The initial version of WannaCry was halted rather quickly when a kill-switch domain was registered and pointed to a sinkhole.

NotPetya

NotPetya is a piece of malware which looks similar in behavior to Petya at first sight. However, in reality, it is not ransomware, since it uses a random key for encryption. This effectively makes files non-restorable, which suggests its purpose is to cause disruption instead of earning ransom. NotPetya mostly spread though a supply chain attack; it was implanted in a software update of a commonly used Ukrainian accounting package. Like WannaCry, NotPetya used the EternalBlue exploit to expand its reach.

Triton

Triton is a piece of malware designed to take down/abuse Industrial Safety Systems (ISS) in a Saudian oil plant. Such ISSs are intended to restore systems to a safe state whenever unsafe operating conditions are detected.
To prevent unintended program changes, such systems have a (physical) key switch, which needs to be set to program mode in order to allow for updates to be performed. Still, oftentimes, this switch is left in program mode to make it easier to perform updates remotely whenever they are necessary.
The eventual aim of the attacker was most likely[1] to cause an unsafe state in the system, or, in other words, to cause physical damage to the plant itself. The attack consisted of several phases:

1. Initial access to IT network. Several backdoors and/or C2 servers were installed.
2. Execute a program which poses as the legitimate engineering software, but which actually executes Python code (compiled with `py2exe`) for executing the next stage of the attack.
3. Execute 'shellcode'
    1. Verify target memory & create a new program (marked with some structure) in the controller.
    2. Implant an installer (bin file included with malicious program).
    3. Execute the installer to implant a backdoor (the other bin file included with the malicious program).
    4. Never actually performed: cause trouble.

The attack, which took place over several years, was eventually discovered because 3 main processes run in parallel. They check whether their memory is consistent from time to time, and (partially) shut down the system if not. After some shutdowns occurred repeatedly, this led to an investigation which discovered the malware.

Mitigations for Triton include connecting the controllers only to necessary networks, switching the controller to program mode only when necessary and signing programs with certificates. Alternatively, monitoring and/or detection could be performed by providing rules which detect the malware, through logging or by performing configuration and network analysis.

Maastricht University ransomware attack

In late 2019, Maastricht University was attacked by ransomware. The attack started with a phishing email that delivered a Microsoft Office file with macros that installed malware. After that, several exploits were employed for lateral movement. Just before Christmas, the actual ransomware attack was deployed, which also disabled antivirus as it did its work. Only Windows domain-joined systems were affected, which included backup systems. Eventually, Maastricht University was 'forced' to pay the ransom.

Colonial Pipeline ransomware attack

In May 2021, Colonial Pipeline was attacked with ransomware. The attackers initially got in through a compromised password. The attack, which quickly led to oil shortages, led to a national emergency in the US.

Equifax hack

In 2017, credit bureau Equifax was hacked. The attackers initially got in through a vulnerability in outdated Apache software, which allowed for direct and opportunistic attacks. This attack led to

---

[1] We do not know this for sure, since the attacker was never able to perform their attack; they were discovered prior to being able to execute their attack.

sensitive data of many US and UK people being leaked. One of the main issues in this case is that the customers of data brokers like Equifax are the companies paying for access to data (and not the individuals whose private data is being stored). This leads to perverse incentives to not care about cybersecurity.