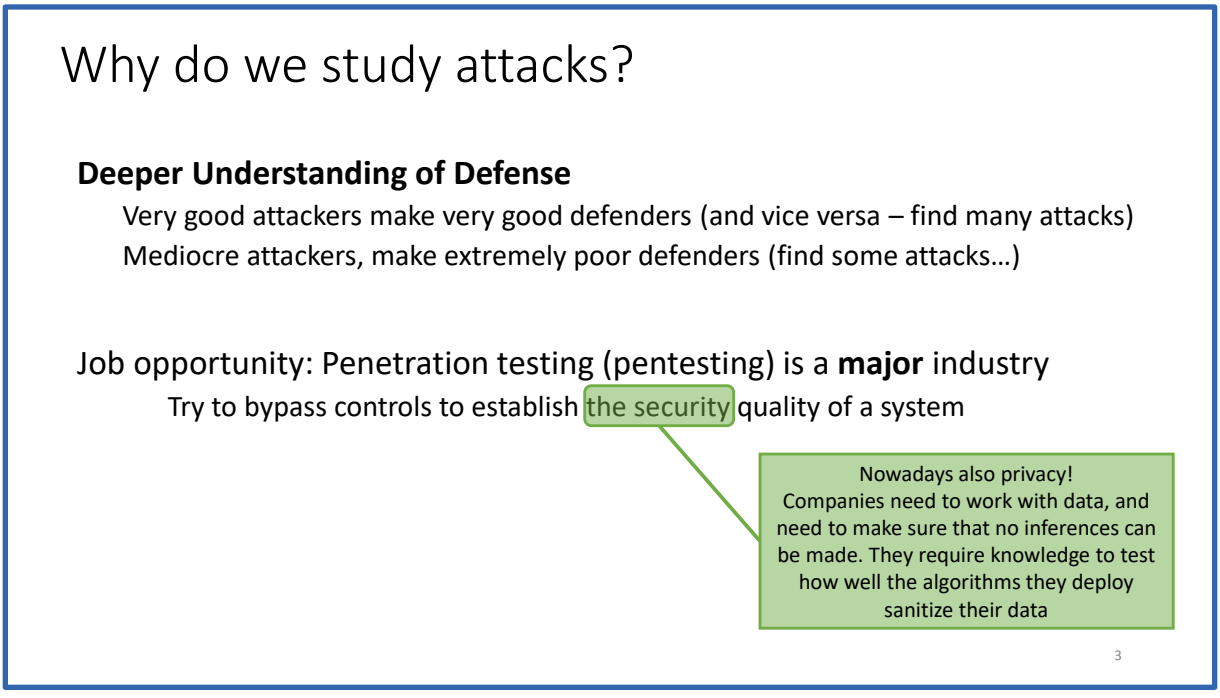
Computer Security (COM-301)

Adversarial thinking
Attacks and defenses

Slides by Carmela Troncoso.
Some slides/ideas adapted from: Emiliano de Cristofaro, Gianluca Stringhini, George Danezis

Structure of the lecture

- Why studying attacks is so important?
- How are attacks developed?
  - Adversarial thinking process
  - Examples on real world systems
- Which attacks should you worry about?
  - Reasoning process: what can go wrong? what not to do?
  - Example attacks on software



In computer security the study of attacks (besides a lot of fun) is the path to better understand threats on systems in order to build better defences.

Besides, from a practical perspective, being a good attacker gives high chances to find a job. Traditional attacks are on systems and software (so-called pentesting). As systems cannot be perfectly secure (as we have seen and we will continue seeing in the next lectures) companies hire adversaries to try to find vulnerabilities. This does not guarantee security ("the universe of bad things" is too big to explore exhaustively) but it helps lowering the risk of attacks by eliminating low-hanging fruits.

Nowadays, not only security attackers are in demand. As privacy becomes more important, both claimed by society (users) and regulation, industry becomes more interested on hiring knowledge related to the evaluation of algorithms to guarantee privacy.

Why do we study attacks?

Deeper Understanding of Defense

Very good attackers make very good defenders (and vice versa – find many attacks)
Mediocre attackers, make extremely poor defenders (find some attacks…)

Job opportunity: Penetration testing (pentesting) is a **major** industry

Try to bypass controls to establish the security quality of a system

Does lack of found attacks guarantee that the system is secure?

No! we can never be sure we have explored the complete attack space

Related concepts: fail safe principle, sanitization

4

But remember that not finding attacks is **not** a guarantee of security. The attack surface is large ant we cannot guarantee that we have tried all possible attacks.

Recall:
fail safe: because we do not know what can go wrong, if something fails go to a state you know is safe.
sanitization: do not try to avoid all the bad things, only allow known good things in your system.

Why do we study attacks?

Deeper Understanding of Defense

Very good attackers make very good defenders (and vice versa – find many attacks)

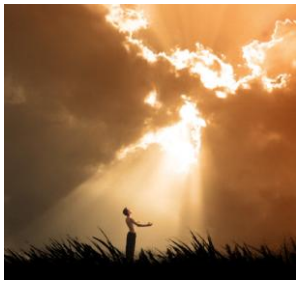
Mediocre attackers, make extremely poor defenders (find some attacks…)

Job opportunity: Penetration testing (pentesting) is a **major** industry

Try to bypass controls to establish the security quality of a system

Remember you cannot freely hack around
Ethics, law, and regulations

How are attacks developed?



6

Attacks (typically) do not happen by chance. It is not that one day one wakes up and has an illumination or an apple falls on your head à la Newton.

Attacks are discovered by studying systems in systematic ways that enable adversaries to explore many angles where there can be a vulnerability.

Remember from Lecture 1.1, that a **vulnerability** is "a specific weakness that could be exploited by adversaries with interest in a lot of different assets"

The security engineering process

1. **Define a security policy (principals, assets, properties) and a threat model.**

2. **Define security mechanisms that support the policy given the threat model.**

3. **Build an implementation that supports / embodies the mechanisms.**

During the first lecture, we established the steps that a security engineer needs to take in order to secure a system:

At a very, very, high level:

- Decide what to protect from whom
- Decide how to protect it
- Implement the protections

The attack engineering process

"inverse" approach – exploits flaws in the security engineering process

1. **Define a security policy (principals, assets, properties) and a threat model.**
   Adversary can exploit
   - Misidentified principals, assets, or properties
   - Capabilities beyond what is considered in threat model
     (more access or more computational/algorithmic capabilities)

2. **Define security mechanisms that support the policy given the threat model.**

3. **Build an implementation that supports / embodies the mechanisms.**

The attack engineering process exploits weaknesses introduced during the security engineering process due to carelessness, lack of knowledge, or errors.

What can go wrong when creating the security policy:
- Forgetting principal, assets, or properties. If any of this is not considered, there may be a valuable asset whose security is not protected from a particular principal.

What can go wrong when deciding the threat model:
- Underestimation of the adversary. One thinks that the adversary has less computational power than in reality (e.g., does not have enough resources to perform denial of service), or one assumes that he knows no algorithm that can break the security policy (e.g., infer secret assets to break confidentiality, compute a hash collision to break integrity).

The attack engineering process

Exploiting misidentified assets in the security policy

EXAMPLE 1 – EXTRACTING KEYS FROM HARDWARE SECURE MODULES (HSMS)

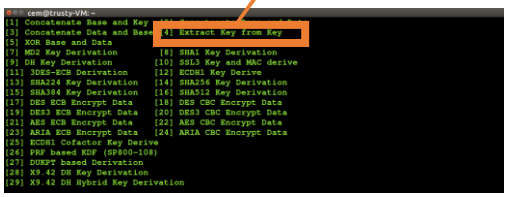
HSMs implement PKCS#11 standard for interoperability

API to create internally a new key from the secret key:

Given bits_length and offset, it uses bits_length of the secret key from position offset

How would you exploit this function?

Create a new key using a substring of an existing key.



https://randomoracle.wordpress.com/2015/08/13/safenet-hsm-key-extraction-vulnerability-part-i/

An HSM is a CPU secured physically. That is, it can hold cryptographic keys that cannot be extracted by observing the device, or measuring the device characteristics (power consumption, computation timing, etc.)

Part of their security comes from having a *strict* API to interact with them. Following economy of mechanism, HSMs can only be accessed through a small set of functions.

One of the functions available is “Extract key from key”. On input offset and key length it *internally* generates a key of the designated length using length bits of the secret key of the HSM starting at position offset.

The attack engineering process

Exploiting misidentified assets in the security policy

PKCS#11 considers the full key an asset to protect, but not bytes of the key

EXAMPLE 1 – EXTRACTING KEYS FROM HARDWARE SECURE MODULES (HSMS)

Assume a strong key exists in the HSM

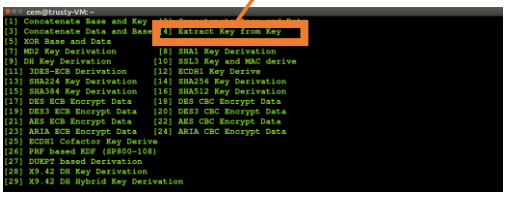
Ask HSM to derive a new key of length 1 byte at offset 0

Use new key to do an operation, say HMAC on a known input
(allowed by the HSM)

Brute force the key
(input known, output known, key only 1 byte)

Repeat with keys at different offsets ⟶ Full key recovery!

Create a new key using a substring of an existing key.



https://randomoracle.wordpress.com/2015/08/13/safenet-hsm-key-extraction-vulnerability-part-i/

This allows a strategic adversary to create small keys (e.g., of size one byte), and ask the HSM to do operations with this key.

As the key to this operation has only 2^8 bits, it is possible to find using exhausting search. By asking for different offsets, the adversary can eventually recover the fill key.

The attack engineering process

Exploiting unforeseen access capabilities

In both cases the adversary had remote access to functionality that was not foreseen by the threat model

EXAMPLE 2 – FROM CABLE TO THE AIR

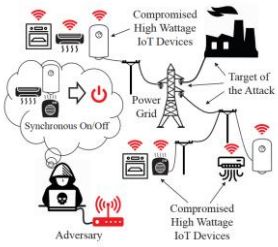
Engine Control Units (ECU) control the vehicle

ECU connected to GSM/WiFi give a remote adversary access to the CAN bus and all the (safety) functions of the vehicle

EXAMPLE 3 – IOT DEVICES ARE A WEAK LINK

IoT weakly protected devices connected to internet

MadIoT - manipulation of demand via IoT
(Princeton U.) – hackers can compromise the Smart Grid with ~100K devices



https://www.forbes.com/sites/andygreenberg/2013/07/24/hackers-reveal-nasty-new-car-attacks-with-me-behind-the-wheel-video/#4b536af4228c
https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/
https://www.ft.com/content/2c17ff5e-4f02-11e8-ac41-759eee1efb74

11

In a car, all instructions are given to different parts using the CAN bus. The CAN bus was never secured, because it was always assumed that an adversary would need physical access to the vehicle in order to read/write from this bus. As electronics advanced, the bus was connected to the Engine Control Unit, but it was still hard to access because one would also need physical access to the vehicle to interact with the ECU. In modern vehicles, however, the ECU is now connected to the internet (via WiFi or GSM) to enable remote updates of the vehicle firmware, or infotainment. As a result, now remote adversaries have access to the most critical part of the vehicle. Hackers have demonstrated that indeed the can gain access and get full control of the vehicle and perform any function: brake, steer, or change sensor readings to make drivers believe their vehicle is not working.

A similar thing happens with Electrical centrals and the power grid. Centrals and distribution centers count with very strong protections. They are typically not connected to the internet, and if they do they have very strong firewalls. Similarly, modifying electricity consumption from traditional infrastructure is hard. One has to have physical access to appliances to switch them on. Other devices connected (computers) are also somehow protected. But nowadays we have (million) more devices connected, the Internet of Things. They are small and many times badly protected because their manufacturers are not well-trained. For instance, many of them still have admin:admin as one of the authorized login/password. Researchers at Princeton University showed that, by accessing these devices, which in turn are connected to the power grit, they can create spike electricity demands that can even bring down regions of the grid.

The attack engineering process

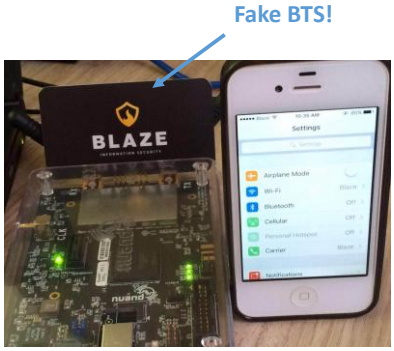
Exploiting unforeseen capabilities

EXAMPLE 3 – UNILATERAL USER AUTHENTICATION IN GSM

When GSM was designed antennas (Base Transceiver Stations - BTS) were difficult to implement and expensive to build.

Thus, operators decided that the network did not need to authenticate!

Nowadays, commodity hardware can be used to fake a base station and perform a man in the middle (eavesdrop, impersonate,…)!



https://wildfire.blazeinfosec.com/practical-attacks-against-gsm-networks-part-1/

Another example of unprotected infrastructure due to underestimating the capabilities of the adversary is the GSM network. When it was created, it was assumed that creating an antenna had prohibitive cost both in materials and know-how. Thus, when running the protocol to connect phones to the network, the antennas do not authenticate.

Nowadays, however, we have commodity Software Defined Radio boards that can easily be programmed to impersonate an antenna. As there is no authentication, it is easy to spoof a network base station and trick mobile phones into connecting to your antenna instead of a real one.

https://iacr.org/submit/files/slides/2023/rwc/rwc2023/3/slides.pdf
https://www.google.com/search?q=https://security.googleblog.com/2023/08/android-14-introduces-first-of-its-kind-cellular-connectivity-security-features.html
https://www.eff.org/deeplinks/2023/09/apple-and-google-are-introducing-new-ways-defeat-cell-site-simulators-it-enough

The attack engineering process

Exploiting unforeseen computational/algorithmic capabilities

EXAMPLE 4 – THE MACHINE LEARNING REVOLUTION

The power of inference at your fingertips!
Apparently irrelevant information becomes critical for the security of the system

Learn to break better and faster!


Help! Hackers Stole My Password Just By Listening To Me Type On Skype!

Thomas Brewster

For many, everyday life involves sitting in front of a computer typing endless emails, presentation documents and reports. Then there's the frequent typing of passwords just to get access to those files. But


The Real First Class? Inferring Confidential Corporate Mergers and Government Relations from Air Traffic Communication

Martin Strohmeier*, Matthew Smith*, Vincent Lenders†, Ivan Martinovic*

*University of Oxford, UK †armasuisse, Switzerland


Using deep learning to break a Captcha system

Using Torch code to break simpleaptcha with 92% accuracy

Captcha is used as a common tactic to stop bots from entering a website. Any visitor to a website is presented with a text image containing some letters which can be read by a human and not by automated bots. They are quite frequently used to stop automated password hacking or automated login to websites etc. The following is taken from the wikipedia page[1] of captcha. As captchas are usually used to deter software programs they can be usually very hard to read and human accuracy can be around 93% [2]. It also takes something like 10 secs to read a captcha. As can be seen this takes quite a toll on the user experience.

Breaking Captchas

There are a few approaches to defeating CAPTCHAs: using cheap human labor to

Machine learning eases attacks, as it simplifies their implementation through substituting complex modeling tasks by data collection

Regarding computation/algorithmic capabilities, we have many examples.

On the capabilities side we have the greater capabilities of state agencies (e.g., NSA) to brute-force RSA long keys. One reason why the internet is changing to longer keys (2048) and mainly Diffie Hellman exchange, as we will see in the network security lecture.

On the algorithmic side, the most paradigmatic example nowadays is the use of machine learning to turn algorithm design into a data-driven problem. Adversaries do not need to think about clever algorithms to infer secret information, or to find ways to fool a detector, but just a lot of data to create a machine learning model that does it for them.

The attack engineering process

Exploiting unforeseen computational/algorithmic capabilities

EXAMPLE 3

The powe

Learn to b

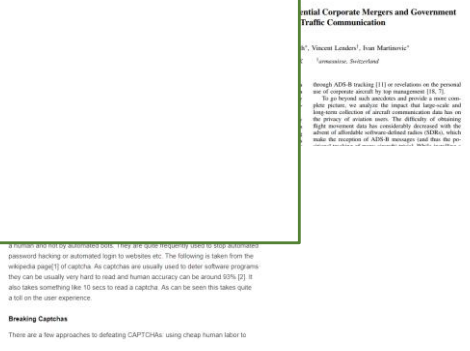
THE MACHINE LEARNING REVOLUTION: ALSO WORKS FOR THE GOOD GUYS!!

Improved malware detection

Predicting zero days (unknown vulnerabilities)

Identifying vulnerable devices

Automated log analysis


ntial Corporate Mergers and Government
Traffic Communication

through ADS-B tracking [11] or revelations on the personal use of corporate aircraft by top management [18, 7].

To go beyond such anecdotes and provide a more complete picture, we analyze the impact that large-scale and long-term collection of aircraft communication data has on the privacy of aviation users. The difficulty of obtaining flight movement data has considerably decreased with the advent of affordable software-defined radios (SDRs), which make the reception of ADS-B messages (and thus the po-

a human and not by automated bots. They are quite frequently used to stop automated password hacking or automated login to websites etc. The following is taken from the wikipedia page[1] of captcha. As captchas are usually used to deter software programs they can be usually very hard to read and human accuracy can be around 93% [2]. It also takes something like 10 secs to read a captcha. As can be seen this takes quite a toll on the user experience.

Breaking Captchas

There are a few approaches to defeating CAPTCHAs: using cheap human labor to

It must also be said that machine learning has also improved the situation from the defender point of view:

- Improve malware detection by being able to process many more features than template-based detectors (see lecture on Malware)

- Improve our capability to detect configurations that are vulnerable (hard to enumerate, but modelable if you have enough data).

- Improve our capability to find malicious complex patterns in logs, even when adversaries try to hide their actions

The attack engineering process

"inverse" approach – exploits flaws in the security engineering process

1. **Define a security policy (principals, assets, properties) and a threat model.**
   Adversary can exploit
   - Misidentified principals, assets, or properties
   - Capabilities beyond what is considered in threat model
     - (access or computational/algorithmic)

2. **Define security mechanisms that support the policy given the threat model.**
   Adversary can exploit
   - Design weaknesses/flaws in the security mechanisms

3. **Build an implementation that supports / embodies the mechanisms.**

A second way of breaking the policy is to inspect the security mechanism to find vulnerabilities that can be exploited.

As we have seen in the lecture there are many decisions to be taken when designing architectures and protocols. It is not hard to make mistakes that open the door to attacks.

The attack engineering process

Exploiting security mechanisms design weaknesses

In both cases the algorithms were secret, but researchers reverse engineered them. Once the algorithms were known researchers identified vulnerabilities that allowed them to decrypt and read messages, and even recover the key.

EXAMPLE 1 – WEAK CRYPTOGRAPHIC PRIMITIVES

Tesla – Key Fob algorithm to start the car allows to recover key in seconds (with pre-computation)

GSM – A5/1 and A5/2 weak allow ciphertext only attacks Can be real time by FPGA parallel computation!

Security by obscurity is a bad idea <- Open design principle!

https://web.archive.org/web/20090821163913/http://reflextor.com/trac/a51/wiki
https://www.securityweek.com/hackers-can-clone-tesla-key-fobs-seconds/
https://www.vice.com/en/article/watch-hackers-steal-a-tesla-model-s-key-fob-hack/

Two examples of weak designs, that used security by obscurity as extra protections.

In both cases the adversaries could reverse engineer the algorithm and then use weak points of the cryptographic design to recover the key.

The attack engineering process

"inverse" approach – exploits flaws in the security engineering process

1. **Define a security policy (principals, assets, properties) and a threat model.**
   Adversary can exploit
   - Misidentified principals, assets, or properties
   - Capabilities beyond what is considered in threat model
     - (access or computational/algorithmic)

2. **Define security mechanisms that support the policy given the threat model.**
   Adversary can exploit
   - Design weaknesses/flaws in the security mechanisms

3. **Build an implementation that supports / embodies the mechanisms.**
   Adversary can exploit
   - Implementation or operation problems that allow you to subvert the mechanisms

After the design comes the implementation. Software is very complex and many times small errors enable the adversaries to infiltrate the TCB (see final slides of this lecture and the software security lecture).

The attack engineering process

Exploiting bad operation decisions to subvert security mechanisms

EXAMPLE 1 – WEP BAD USE OF RC4

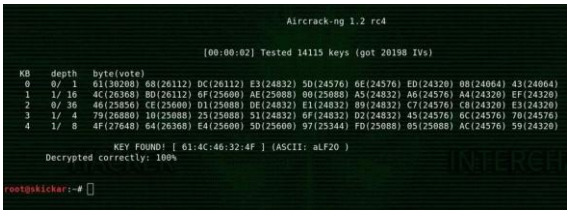
WEP uses RC4, a secure stream cipher when the IV is random.

In WEP the IV is defined to have 24 bit. The implementation uses this 24 bits in such a way that the IV is repeated every 5000 / 6000 frames!

Adversary can accelerate the attack by spoofing MAC addresses to ask for more frames

When the IV is repeated, the stream produced by RC4 that is XORed with messages is repeated. This effectively is a repeated One Time Pad, and thus allows to recover messages. Because of some particularities of how RC4 is constructed, one can even recover the secret key.

Can be also seen as the WEP protocol is a flawed design



https://null-byte.wonderhowto.com/how-to/hack-wi-fi-cracking-wep-passwords-with-aircrack-ng-0147340/
https://asecuritysite.com/encryption/rc4_wep

One implementation problem was the decision of using RC4 (a secure stream cipher) with a small IV for WEP (a protocol to encrypt WiFi communications). The fact that the IV is very small means that if there is enough traffic the IV will be repeated. And remember that if we repeat the IV with a given key, a stream cipher will output the same pseudorandom string! (effectively we are doing the same as repeating the use of a one time pad).

Given this reuse of a stream, adversaries can recover information about the message, and because of the internals of RC4, even recover the symmetric key and decrypt the full communication.

The attack engineering process

Exploiting implementation flaws to subvert security mechanisms

EXAMPLE 2 – BUGS, BUGS AND MORE BUGS

Programmers make mistakes:

- They forget checks, or check the wrong things
- They do not sanitize, or do not sanitize correctly
- They forget to protect what needs to be protected
- They get confused about origin or reliability of data / variables (Ambient authority & confused deputy)

Sudo Flaw Lets Linux Users Run Commands As Root Even When They're Restricted

October 14, 2019 Mohit Kumar

19

Beside bad parametrization of algorithms in implementations, programmers also make mistakes. These mistakes, often known as bugs, become vulnerabilities that enable the adversary to perform actions the break the security policy.

Why does this work?

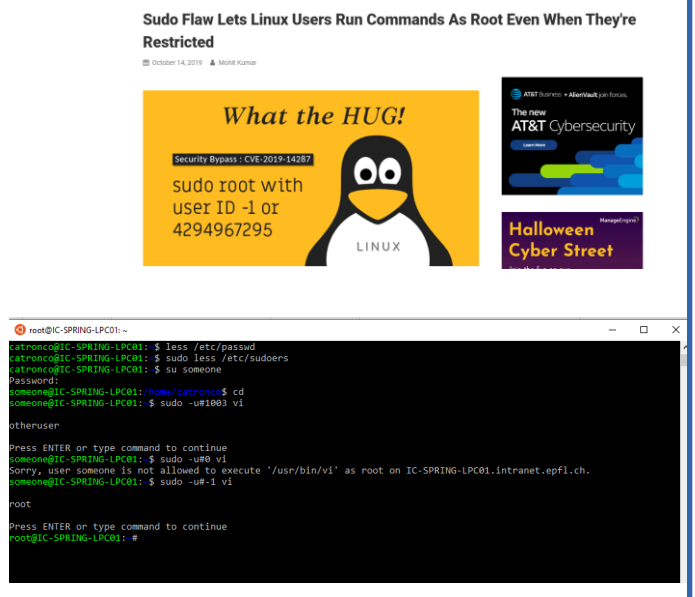
Sudo program uses two routines, one of them does the change in UID (Remember UID is what determines the permissions of the program).

That routine understands "-1" as "do nothing".

Because the routine is called inside sudo, which is being executed as root (UID = 0), then the program comes out without changing, and stays with the same UID.

Bright side, only exploitable under certain configurations in which users can execute sudo on potential dangerous programs for some users except for root:

```
someone ALL=(ALL, !root) /usr/bin/vi
```



An example of a bug that allows users to gain root access in Linux is a wrong check in the implementation of the sudo command. In a nutshell, the problem was that:

- the sudo command runs as root (UID = 0)
- When given a user UID to run a command as another user, during the execution of sudo the system changes the UID to that user
- Because of a wrong comparison, when the UID provided is -1, this change does not happened, so the program returns from sudo keeping UID=0, i.e., keeping root privileges.

[Interestingly, it also works with 4294967295 because it is the unsigned version of -1]

This bug only works when there is a particular configuration of who can run which programs using sudo. As a result, even though it is easy to launch this attack, it has very little impact.

In any case, you should update your Linux distribution!

Computer Security (COM-301)

Adversarial thinking
Reasoning as a defender - I

Slides by Carmela Troncoso.
Some slides/ideas adapted from: Emiliano de Cristofaro, Gianluca Stringhini, George Danezis

Reasoning about attacks
Threat modelling methodologies

IDEA: help security engineers reason about threats to a system - **"What can go wrong?"**

Threat modelling

Process to identify potential threats and unprotected resources with the goal of prioritizing problems to implement security mechanisms.

Systematic analysis:

- What are the most relevant threats?
- What kind of problems can threats result on?
- Where should I put more effort to protect?

22

In order to defend the system from attacks, it is also important to reason about them in a systematic way. In other words, explore the ways in which things can wrong for your security policy.

Threat modelling methodologies aim at providing guidance or security teams to explore risks in a systematic way

Reasoning about attacks
Threat modelling methodologies

IDEA: help security engineers reason about threats to a system - **"What can go wrong?"**

Attack trees

The attack goal is the root, and the ways to achieve this goal are represented by the branches. The leafs are the weak resources.

STRIDE

Identify system entities, events, and the boundaries of the system.

Reason about threats enumerating the type of threats that can be embodied by the adversary

P.A.S.T.A.

Start from business goals, processes, and use cases.

Find threats within business model, assess impact, and prioritize based on risk

Many more!

23

There are many threat modelling methodogies. There is no "order" or "recommendation". They are different and one should choose the one that seems the most adequate for their use case

Reasoning about attacks
STRIDE (by Microsoft)

Model the target system, with entities, assets, and flows. Then reason about:

Threat	Property threatened	Example
Spoofing	Authenticity	A member of the council of Ricks convinces Morty that he is the real Rick
Tampering	Integrity	The bad minion modifies the plan message send by Gru to our favorite minion Bob
Repudiation	Non-repudiability	Summer denies having told Morty that Rick was waiting for him
Information disclosure	Confidentiality	Summer learns about the secret plans of Rick and Morty
Denial of Service	Availability	The minions flood Dr. Nefario's lab with bananas and he cannot receive the latest weapons
Elevation of Privilege	Authorization	Bob the minion gains access to the system with Gru's credentials

24

Microsoft employees propose the methodology STRIDE to explore possible threats (and security properties that can be broken).

Starting from a description of the system, STRIDE helps the analyst consider different dangers and harms in a systematic way.

Reasoning about attacks
Brainstorming using cards

http://securitycards.cs.washington.edu/index.html

25

Games can help! This is an example of several card-based games that help brainstorming about threats and vulnerabilities in order to explore all possible attack vectors and consequences in a system.