Lecture 1 - Introduction

Assets to protect in computer systems

- **Confidentiality** - refers to protecting information from being accessed by unauthorized parties. A failure to maintain confidentiality means that someone who shouldn't have access has managed to get it, through intentional behavior or by accident.
- **Integrity** - refers to ensuring the authenticity of information. Which means, that information is not altered, and that the source of the information is genuine.
- **Availability** - means that information is accessible by authorized users. If an attacker is not able to compromise the first two elements of information security (see above) they may try to execute attacks like denial of service that would bring down the server, making the website unavailable to legitimate users due to lack of availability.

Importance of policy, difference of policy and mechanism

How to ensure above aspects ? By defining policies.

A **security policy** is a statement of what is, and what is not, allowed.
In practice, they normally describe in English what users and staff are allowed to do. The ambiguity inherent in such a description leads to states that are not classified as “allowed” or “disallowed.”

A **security mechanism** is a method, tool, or procedure enforcing a security policy.
It is important to decide, which aspect will be the main one, because in the real systems is usually not possible to ensure security in all three aspects. It’s also complicated to come up with new rules and guidelines, because we have to guess our assumptions and future impacts on the system.

Security Economics

Usually companies invest into the security the minimum amount required -> therefore systems won’t be ever totally safe, because it is expensive.

The companies are usually fixed on one supplier. Which means that the **technical lock-in** comes to play. It’s expensive to transfer whole company to new supplier, which means that most companies are locked to one supplier and takes what the supplier provides.

While using nowadays systems, we have **asymmetric information**, we don’t know how well is the system protected. Therefore if the two systems have different price, we logically choose the one which is cheaper. (for more info https://en.wikipedia.org/wiki/The_Market_for_Lemons)

Also, if my system should succeed on the market, it has to the first system on the market, which means no time for security stuff.

Openned vs Closed system

Open system - source code is available.
Closed system - source code is not available.

It was proved that, under the standard assumptions of reliability growth theory, it does not matter whether the system is open or closed. Opening a system enables the attacker to discover vulnerabilities more quickly, but it helps the defenders exactly as much. -> But still this works in theory in ideal world. In the real world it depends on many aspects.

Open system – faster detection of errors from both sides, developer and attacker
Closed system – the opposite of the open system, for both sides it's difficult to find security leaks.

Goals of Security

- **Prevention**
  - Typically, prevention involves implementation of mechanisms that users cannot override and that are trusted to be implemented in a correct, unalterable way, so that the attacker cannot defeat the mechanism by changing it.
  - Simplest way of prevention: passwords
- **Detection**
  - Detection mechanisms accept that an attack will occur; the goal is to determine that an attack is underway, or has occurred, and report it.
- **Recovery**
  - to stop an attack and to assess and repair any damage caused by that attack.
  - the system continues to function correctly while an attack is underway

Lecture 2 - Guidelines for writing Secure Code

The danger of Turing complete languages.

An exploit that provides a Turing-complete execution environment allows the attacker to run any algorithm they wish. They will not be limited by the number of available instructions.

A system of data manipulation rules (script, program, instruction set, etc) is considered Turing-complete if it can be used to simulate a Turing machine. In other words, if someone can execute Turing-complete code on a vulnerable computer, they can compute anything, barring resource limitations.

Imagine if you had access to a shell, running as root. Obviously root can do pretty much everything, but what if the only things you can do on the shell is run echo, id, and cd? No if statements, no mv, no sed... You'd be pretty limited in what you can do. That scenario would not be Turing-complete because you do not have enough commands to make it turing-complete.

You cannot determine if the program will stop or not. Therefore you cannot verify it.

Managing a security project

We start to create the security of the system in the time of designing the system.

During the system security design the designers don't know where the attack may come from.

The design starts with defining the **policies** and their counterpart **mechanisms**. Also, the developers define following questions:

- Who has the access to individual parts of the system?
- Where are the critical data?
- Who has the rights to influence the critical data?
- Where the attacker may come from?
- How crucial are the risks and if it worth to deal with them?

Development models and their impact on design of secure systems

Waterfall

+ The system is fully defined at the beginning of the process.
+ If the threats are properly detected at the very beginning, there won't be new threats in the future.
+ It is cheaper to mitigate threat at the beginning

- If the threats are not properly detected at the very beginning. It's complicated to later modify the SW.
- If we choose technology at the beginning and during the development the technology outlives its hard to switch to new technology

Iterative (Spiral model)

+ More iterations, designers can react to the changes.

- It is very likely that new security leaks will be brought to the system when in later iterations
- Sometimes it is hard to add security feature in the middle of the system development

Ideal is iterative model with larger phases, each phase is made in waterfall fashion.
Other models also used are **V-model** and **W-model**.

Steps in threat modelling

1. Decompose the application.
2. Determine threats.
    - **STRIDE**
        - **S**poofing the identity
        - **T**ampering with the data – unauthorized data modification
        - **R**epudiation - not controlling user actions enough, in the system where repudiation is not ensured, we cannot say who performed actions on given account
        - **I**nformation disclosure - is when an application fails to properly protect sensitive and confidential information from parties that are not supposed to have access to the subject matter in normal circumstances.
        - **D**enial of service
        - **E**levation of privileges
3. Rank the threats by decreasing risk.
    - **DREAD**
        - **D**amage potential - how bad would an attack be?
        - **R**eproducibility of the attack - how easy is it to reproduce the attack?
        - **E**xploitability - how much work is it to launch the attack?
        - portion of **A**ffected users - how many people will be impacted?
        - **D**iscoverability - how easy is it to discover the threat?
4. Chose how to respond to threats (mitigate).
    - Do nothing
    - Warn the user
    - Remove the problem (feature)
    - Fix the problem
5. Chose techniques to mitigate the threats.

Common guidelines and practices of coding and designing systems

Classification of threats

- Identity theft
- Data corruption – modification of the database/ code of the application etc.
- Performing a user cation followed be denial of that action – example: Take of package than deny the takeover.
- Information leak
- Denial of service
- Obtaining privileges

Secure

- **by design:**
  - All people developing the product should be aware of security.
  - Have a person responsible for security.
  - Have coding guidelines minimizing dangerous code.
  - Fix all flaws as soon as possible.
  - Make everything as simple as possible
- **by default:**
  - Do not install all features by default.
  - Use secure mechanisms to protect critical resources.
  - Use least privilege principle.
- **in deployment:**
  - The product is maintainable after install.
  - Possible of update to face new threats.
  - Security features can be administered.
  - Create good quality patches as soon as possible.
  - Inform user how to use product securely.

Security principles

- Use defense in depth - **Defense in depth** (also known as **Castle Approach**[1]) is an information assurance concept in which multiple layers of security controls.
- Use least privilege
- Psychological acceptability
- Learn from mistakes
- Minimize your attack surface
- Separation of privileges
- Assume external systems are insecure
- Backward compatibility always
- Plan on failure & fail securely
- Security features != secure features
- Never depend on security through obscurity alone
- Don't mix code and data

A **bastion host** is a special-purpose computer on a network specifically designed and configured to withstand attacks. The computer generally hosts a single application, for example a proxy server, and all other services are removed or limited to reduce the threat to the computer.

Lecture 3 – Covert channels, steganography and steganalysis

Covert channel

Cover channel is an information flow mechanism within a system that is based on the use of system resources not normally intended for communication between the users of the system. In covert channel, there are two parties who **want** to communicate together against the policy.

- **covert storage channel**
  - use system variables and attributes (other than time) to signal information, e.g. — file status
  - communication through storing/reading data
- **covert timing channel**
  - uses a temporal or ordering relationship among accesses to a shared resource
  - Example: delays between packets transmitted over computer networks

Example:
High privilege system -> own file name passwords.txt
Low privilege system user wants to create the same file (passwords.txt)
This file already exists.
If the LPS allows to create the file, the original is overwritten.
If the LPS denies to create the file, it passes the information that the file exists -> covert channel

Identification of covert channels

- Identify all shared resources.
- Identify resources whose visibility is against policy.
- Estimate the bandwidth of the channel. (If the bandwidth is low, it might not be economic to fix the covert channel, because it is not possible to transfer any reasonable amount of data in the reasonable amount of time.)

Side channel attacks

In computer security, a side-channel attack is any attack based on information gained from the implementation of a computer system, rather than weaknesses in the implemented algorithm itself (e.g. cryptanalysis and software bugs). In side channel attacks there are two parties but one does **not want** to communicate. For this attack the communication goes through the channel which is not designed for communication like: timing information, power consumption, electromagnetic leaks or even sound can provide an extra source of information, which can be exploited.

Data remanence - Data may endure for example disc erasure -> proper ways of destroying disks needed
Differential fault analysis - Data may become visible when the system is put in extreme conditions : overheat, high voltage, ...

Example:
Password detection based on evaluation time
Let's say that the password to the system is PASSWORD (8 characters)
If the attacker wants to check all combinations its 26^8 combination.
If the system is designed in a way, that password check is stopped when reading a wrong char.
Than if the attacker is able to evaluate time of the password check, he is easily able to guess the password. He needs only 8*26 combinations.

Steganography

Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video.
The advantage of steganography over cryptography alone is that the intended secret message does not attract attention to itself as an object of scrutiny. Plainly visible encrypted messages, no matter how unbreakable they are, arouse interest and may in themselves be incriminating in countries where encryption is illegal.

Media files are ideal for steganographic transmission because of their large size. For example, a sender might start with an innocuous image file and adjust the color of every hundredth pixel to correspond to a letter in the alphabet. The change is so subtle that someone who is not specifically looking for it is unlikely to notice the change.

Steganographic channel = channel that enables the exchange of the “innocuous” messages.

Steganalysis

the study of detecting messages hidden using steganography; this is analogous to cryptanalysis applied to cryptography.

The problem is generally handled with statistical analysis. A set of unmodified files of the same type, and ideally from the same source (for example, the same model of digital camera, or if possible, the same digital camera; digital audio from a CD MP3 files have been "ripped" from; etc.) as the set being inspected, are analyzed for various statistics. Some of these are as simple as spectrum analysis, but since most image and audio files these days are compressed with lossy compression algorithms, such as JPEG and MP3, they also attempt to look for inconsistencies in the way this data has been compressed. For example, a common artifact in JPEG compression is "edge ringing", where high-frequency components (such as the high-contrast edges of black text on a white background) distort neighboring pixels. This distortion is predictable, and simple steganographic encoding algorithms will produce artifacts that are detectably unlikely.

In some cases, such as when only a single image is available, more complicated analysis techniques may be required. In general, steganography attempts to make distortion to the carrier indistinguishable from the carrier's noise floor. In practice, however, this is often improperly simplified to deciding to make the modifications to the carrier resemble white noise as closely as possible, rather than analyzing, modeling, and then consistently emulating the actual noise characteristics of the carrier. In particular, many simple steganographic systems simply modify the least-significant bit (LSB) of a sample; this causes the modified samples to have not only different noise profiles than unmodified samples, but also for their LSBs to have different noise profiles than could be expected from analysis of their higher-order bits, which will still show some amount of noise. Such LSB-only modification can be detected with appropriate algorithms, in some cases detecting encoding densities as low as 1% with reasonable reliability.

Watermarking

A digital watermark is a kind of marker covertly embedded in a noise-tolerant signal such as audio, video or image data. It is typically used to identify ownership of the copyright of such signal. "Watermarking" is the process of hiding digital information in a carrier signal; the hidden information should, but does not need to, contain a relation to the carrier signal. Digital watermarks may be used to verify the authenticity or integrity of the carrier signal or to show the identity of its owners. It is prominently used for tracing copyright infringements and for banknote authentication.

Characteristics

- robust against distortion / removal attacks.
- its presence can be detected
- it usually has low capacity

RSA (**Rivest–Shamir–Adleman**) is one of the first public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and it is different from the decryption key which is kept secret (private). In RSA, this asymmetry is based on the practical difficulty of the factorization of the product of two large prime numbers, the "factoring problem".

A basic principle behind RSA is the observation that it is practical to find three very large positive integers e, d and n such that with modular exponentiation for all integers m (with $0 \leq m < n$):

$$(m^e)^d \equiv m \pmod{n}$$

and that even knowing e and n or even m it can be extremely difficult to find d. The triple bar (≡) here denotes modular congruence.

Lecture 4 - Access control rights

Security state, security policy, secure system, security model

Let's assume we wrote a good and clear policy. Now we need to find an ideal mechanism to enforce our policy. We need to answer questions, how to find such mechanism and how to prove that it will ensure our policy. Security policy should clearly divide the system into two parts (without intersection) showing when the system is safe and when it isn't.

Security model is a scheme for specifying and enforcing security policies.
Security model provides a formal representation of the access control security policy and its working. The formalization allows the proof of properties on the security provided by the access control system being designed.

Secure system - system, that starts in an authorized state and cannot enter an unauthorized state.

Security state - a subset of states of the system that is related to the security.

State transition - occurs when a command changes the state of the system.

Access control models:
Access control model is a method which should prove that if we start in safe mode, we can't end up in unsafe mode. Unfortunately it is not possible to prove it even if we have a very good policy. Keep in mind that completely safe systems can be also unusable, for example: we put the system into the box which is unreachable from the outside.

Types of access control

- **Mandatory access control (MAC)**
    - When a system mechanism controls access to an object and an individual user cannot alter that access.
    - MAC takes a hierarchical approach to controlling access to resources. Under a MAC enforced environment access to all resource objects (such as data files) is controlled by settings defined by the system administrator. As such, all access to resource objects is strictly controlled by the operating system based on system administrator configured settings. It is not possible under MAC enforcement for users to change the access control of a resource.
- **Discretionary access control (DAC)**
    - If an individual user can set an access control mechanism to allow or deny access to an object, that mechanism is a discretionary access. It bases access rights on identity of the subject and the object involved.
    - Unlike MAC where access to system resources is controlled by the operating system (under **the control of a system** administrator), DAC allows each user to control access to their own data. DAC is typically the default access control mechanism for most desktop operating systems.

Discretionary access control (DAC)

a) **Access control matrix -** a method to precisely describe security state. It contains all the objects and all the subjects and defines the allowed interactions of subjects with objects.

	objects			
subjects	file a	file b	process a	process b
process a	rwo	rw	rwx	rwx
process b	r	rw	r	rw
alice	—	rwo	rwx	rx
bob	rwo	—	rwx	rx

b) **Access control list -** is a list of permissions attached to an object. An ACL specifies which users or system processes are granted access to objects, as well as what operations are allowed on given objects. For instance, if a file object has an ACL that contains `(Alice: read, write; Bob: read)`, this would give Alice permission to read and write the file and Bob to only read it.

Alternative to **access control matrix** is **token matrix**. The problem with security in this model is that subject can share or borrow the token. This model is useful for the server services. More complex access control methods are feasible for example for the military use are **Mandatory access control**, **Role-based access control** and **Multilateral** security.

Mandatory access control (MAC)

Multilateral Security vs Multilayer security

Means the security on the same level of access. In multilateral systems, instead of the information flow-control boundaries being horizontal, as in the Bell-LaPadula model (Figure 8.1) we instead need the boundaries to be mostly vertical, as shown in Figure 8.2. Example: One patient in hospital should not have access to records of another patient.

Figure 8.1 Multilevel security.

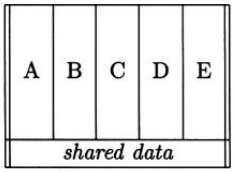


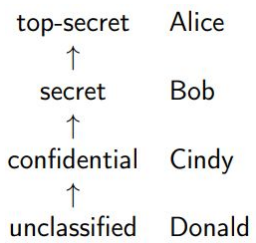
Figure 8.2 Multilateral security.

Multilateral security uses two basic models to implement access control in multilateral security: Compartments and Chinese Wall model.

A) Multilayer security

1. Bell-LaPadulla model for confidentiality

based on hierarchy of access control rights.
Simple Security Condition: S can read O if and only if l(O) < l(S) and S has discretionary read access to O.
Problem of the model -> Bubble up
If Alice modifies the document of Donald, the document is labeled top secret, aka only Alice can access it. In the time, all the documents become top-secret.

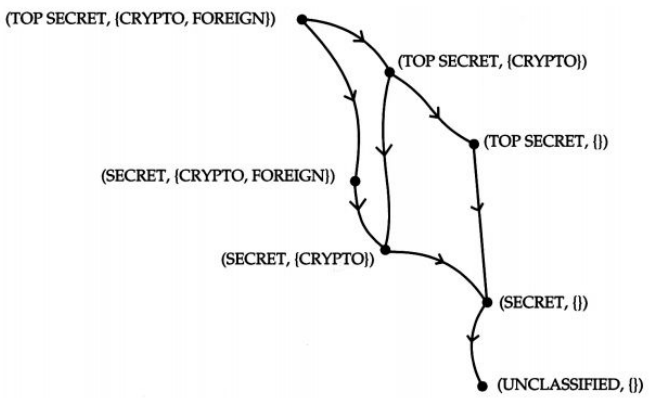


2. Compartments (oddělení)

Compartmentalization, in information security, whether public or private, is the limiting of access to information to persons or other entities on a need-to-know basis to perform certain tasks. In the compartment method individuals has access to resources, which belongs to their code word level.

Example:

The document which has compartments: TOP SECRET, CRYPTO, FOREIGN can read only person which was provided these three code words.



3. The Biba model for integrity

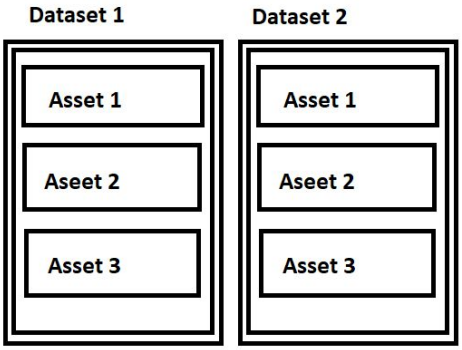
Deals with integrity alone and ignores confidentiality

Subjects can read objects of higher security level but can't write to them. Higher security levels can write to the files of lower security level.

B) Multilateral security

Chinese wall model

In the Chinese wall model, the data of the company are placed to datasets. Each employee has access to defined datasets.



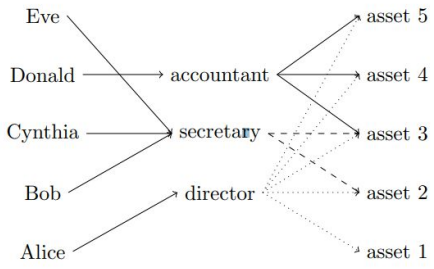
Subject S, can from his dataset be granted access to an object (asset) o only if the object o:

- is in the same dataset of S
- is in dataset which has no access conflicts with the dataset of S

This model works until attackers don't cooperate. Let's say that sharing between datasets A, B is forbidden. But between A, C and B, C is allowed. Then the two employees can pass data through dataset C.

Role-based access control

Provides a more general framework for mandatory access control than BLP in which access decisions don't depend on usernames but on the functions which they are currently performing within the organization. Transactions which may be performed by holders of a given role are specified, then mechanisms for granting membership of a role (including delegation).



Role-based vs Bell-LaPadulla

Role-based system does not have strict hierarchy. User on the level accountant don't have the same access rights as programmer although they are on the same level of access control

Clark&Wilson model

- for integrity Authentication All users has to authenticate before using the system.
- Audit All changes of data are logged such that they can be undone.
- Well-formed transactions All data manipulations must lead from consistent to consistent state.
- Separation of duty The allows each user to run only those programs that reflect her working duty.

Why we are stuck with discretionary access control

People are used to it. We want control over our data.

Lecture 5 – How privileges are enforced in OS?

Privilege escalation

- Vertical escalation -> unauthorized obtaining higher privileges
- Horizontal escalation -> obtaining privileges of the users on the same access level

Reasons why is it hard to prevent privilege escalation?

a) Social engineering - impossible to remove
b) Complete mediation - letting control over process with higher privileges
c) Bugs in code - always there
d) Bugs in hardware - always there

What is needed to prevent privilege escalation?

- **Least privilege principle -** A subject should be given only those privileges that it needs in order to complete its task.
- **Privilege separation -** a technique in which a program is divided into parts with specific privileges. Each part obtains only enough privileges to perform his tasks. This is used to mitigate the potential damage of a computer security vulnerability.

  A common method to implement privilege separation is to have a computer program fork into two processes. The main program drops privileges, and the smaller program keeps privileges in order to perform a certain task. The two halves then communicate via a socket pair. Thus, any successful attack against the larger program will gain minimal access, even though the pair of programs will be capable of performing privileged operations.

- **Use effective monitoring system, to detect unusual actions and blocks them immediately.**

Principle of Complete Mediation

All accesses to objects should be checked to ensure that they are allowed.

This principle restricts the caching of information, which often leads to simpler implementations of mechanisms. Every time that someone tries to access an object, the system should authenticate the privileges associated with that subject. What happens in most systems is that those privileges are cached away for later use. The subject's privileges are authenticated once at the initial access. For subsequent accesses the system assumes that the same privileges are enforce for that subject and object. This may or may not be the case. The operating system should mediate all and every access to an object.

Time of check to Time of use

In software development, time-of-check to time-of-use is a class of software bugs caused by a race condition involving the checking of the state of a part of a system (such as a security credential) and the use of the results of that check.

Victim	Attacker

`if (access("file", W_OK) != 0) {`
`  exit(1);`
`}`
 	
`fd = open("file", O_WRONLY);`
`// Actually writing over`
`/etc/passwd`
`write(fd, buffer,`
`sizeof(buffer));`
	`//`
	`//`
	`// After the access check`
	`symlink("/etc/passwd", "file");`
	`// Before the open, "file" points to the password`
	`database`
	`//`
	`//`

The system confirms that the access to “file” is ok, the attacker creates symbolic link though file into password file and voila he has all the passwords.

Trusted computer base

of a computer system is the set of all hardware, firmware, and/or software components that are critical to its security, in the sense that bugs or vulnerabilities occurring inside the TCB might jeopardize the security properties of the entire system.
The security of the system is based on trusted computer base. You have to trust that its secure.
Our goal is to minimize the size of trusted computer base.
Parts of trusted computer base

- Kernel and all its modules (device drivers)
- Window management systems
- Systems verifying authenticity (SSH, login).
- all root processes ps -ax -u root

Reference(Security) monitor and requirements on it

Security monitoring, sometimes involves collecting and analyzing information to detect suspicious behavior or unauthorized system changes on your network, defining which types of behavior should trigger alerts, and taking action on alerts as needed.

Security monitor performs security monitoring.

Properties of reference monitor

- Non-bypassable
- Evaluable
- Always invoked - always used
- Tamper-proof - nezměnitelný, nelze jej přepsat

Essential services provided by an operating system

OS scheduling

- The activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy (Round Robin, FIFO, priority).
- Context Switch
- The mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time.
    - Example: suspend process when it is waiting for the I/O

Process isolation

- Set of different hardware and software technologies designed to protect each process from other processes on the operating system.
- Example: virtual address space, limited inter-process communication

Inter Process Communication

- A mechanism which allows processes to communicate with each other and synchronize their actions.
- Two ways:
    - shared memory
    - message passing via kernel

Secure OS

Secure OS must have all of the following features:

- Complete mediation – The reference monitor correctly evaluates all paths from processes to syscalls and that it complies with our policy.
- Tamper Proof – Resistant against override - ex. Isolating kernel via protection rings
- Verifiable – formally provable security

Micro vs Monolithic kernel

Monolithic kernel

- A single large process running entirely in a single address space. It is a single static binary file. All kernel services exist and execute in the kernel address space. The kernel can invoke functions directly.
- EVERYTHING IS IN KERNEL
- Examples: Unix, Linux

Microkernel

- The kernel is broken down into separate processes, known as servers. Some of the servers run in kernel space and some run in user-space. All servers are kept separate and run in different address spaces. Servers invoke "services" from each other by sending messages via IPC (Interprocess Communication). This separation has the advantage that if one server fails, other servers can still work efficiently.
- Only the very important parts like IPC(Inter process Communication), basic scheduler, basic memory handling, basic I/O primitives etc., are put into the kernel. (Kernel space)
- Slower than monolithic because of the additional message passing
- Examples: Mac OS X, Windows NT

Mechanisms of contemporary operating systems to isolate processes and prevent compromise of itself

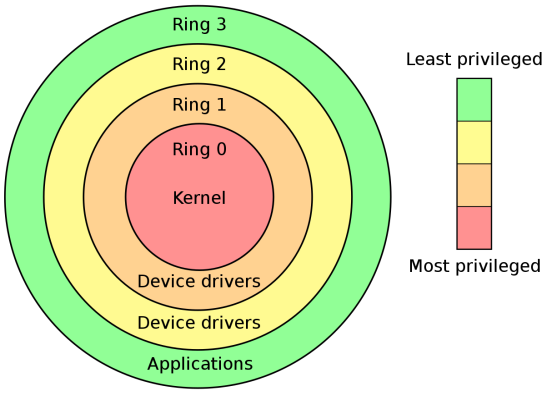
Virtual memory management

- Allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.
- Control bits: Read/Write, User/Supervisor, Accessed, Dirty, present

System call

- First, the user application program sets up the arguments for the system call.
- After the arguments are all set up, the program executes the "system call" instruction.
- This instruction causes an exception: an event that causes the processor to jump to a new address and start executing the code there.
- The instructions at the new address save your user program's state, figure out what system call you want, call the function in the kernel that implements that system call, restores your user program state, and returns control back to the user program.

OS rings



- x86 has 4 rings (0 - 3)
- other CPUs only have two modes (supervisor and user)
- Linux uses only ring 0 and 3
- ring 0
    - level with the most privileges and interacts most directly with the physical hardware such as the CPU and memory.
    - contains kernel
- ring 1,2 (supervisor level)
    - designed for drivers
    - because paging only has a concept of privileged (ring 0,1,2) and unprivileged, the benefit to rings 1 and 2 decreased greatly
    - can access supervisor pages, but if they attempt to use a privileged instruction, they still GPF (general protection fault) like ring 3 would
    - VirtualBox runs the guest kernel code in ring 1
- ring 3 (user mode)
    - cannot change its own ring
    - cannot modify the page tables (can't see memory of other processes)
    - cannot register interrupt handlers
    - cannot do IO instructions (don't have arbitrary hardware accesses)
    - if process wants the kernel to do something it generates an interrupt and the handler will decide if the kernel will allow this action
- ring -1
    - further capabilities than ring 0
    - for CPUs which support Virtualization, this is where the hypervisor is running
        - CPUs without support for virtualization run the VMM in ring 0 and guest OS in ring 3
- major advantages are that it's easier to make programs as we are more certain that one won't interfere with the other

Special gates between rings are provided to allow an outer ring to access an inner ring's resources in a predefined manner, as opposed to allowing arbitrary usage. Correctly gating access between rings can improve security by preventing programs from one ring or privilege level from misusing resources intended for programs in another. For example, spyware running as a user program in Ring 3 should be prevented from turning on a web camera without informing the user, since hardware access should be a Ring 1 function reserved for device drivers. Programs such as web browsers running in higher numbered rings must request access to the network, a resource restricted to a lower numbered ring.

Well known attack vectors

Buffer overflow

- Buffer overflow is a term which indicates that a buffer(a memory unit) exceeds/overflows it boundaries and overwrites adjacent memory locations . (Example: Array overflow, Stack overflow, Heap overflow)
- By sending in data designed to cause a buffer overflow, it is possible to write into areas known to hold executable code and replace it with malicious code, or to selectively overwrite data pertaining to the program's state, therefore causing behavior that was not intended by the original programmer.
- Protection: Canaries

Stack vs Heap:
The stack is the memory set aside as scratch space for a thread of execution. When a function is called, a block is reserved on the top of the stack for local variables and some bookkeeping data. When that function returns, the block becomes unused and can be used the next time a function is called. The stack is always reserved in a LIFO order; the most recently reserved block is always the next block to be freed. This makes it really simple to keep track of the stack; freeing a block from the stack is nothing more than adjusting one pointer.

The heap is memory set aside for dynamic allocation. Unlike the stack, there's no enforced pattern to the allocation and deallocation of blocks from the heap; you can allocate a block at any time and free it at any time. This makes it much more complex to keep track of which parts of the heap are allocated or free at any given time; there are many custom heap allocators available to tune heap performance for different usage patterns.

Stack overflow

- Stack overflow happens when program overflow is memory block assigned for his stack. Which means that he gets to block of memory (stack) of another program. And can execute commands from his memory space, which means with his privileges.

Heap overflow

- type of buffer overflow that occurs in the heap data area
- Exploitation is performed by corrupting this data in specific ways to cause the application to overwrite internal structures such as linked list pointers.

Structured Exception Handler Overwrite

- Structured exception handling (SEH) is an exception handling mechanism included in most programs to make them robust and reliable. It is used to handle many types of errors and any exceptions that arise during the normal execution of an application. SEH exploits happen when the exception handler of an application is manipulated, causing it to force an application to close. Hackers normally attack the logic of the SEH, causing it to correct nonexistent errors and lead a system to a graceful shutdown. This technique is sometimes used with buffer overflows to ensure that a system brought down by overflows is closed to prevent unnecessary and excessive damage.

Return oriented programming

- Return-oriented programming is a computer security exploit technique that allows an attacker to execute code in the presence of security defenses such as executable space protection and code signing.
- In this technique, an attacker gains control of the call stack to hijack program control flow and then executes carefully chosen machine instruction sequences that are already present in the machine's memory, called "gadgets". Each gadget typically ends in a return instruction and is located in a subroutine within the existing program and/or shared library code. Chained together, these gadgets allow an attacker to perform arbitrary operations on a machine employing defenses that thwart simpler attacks.

General defenses to stack / heap overflow and structured exception handler override

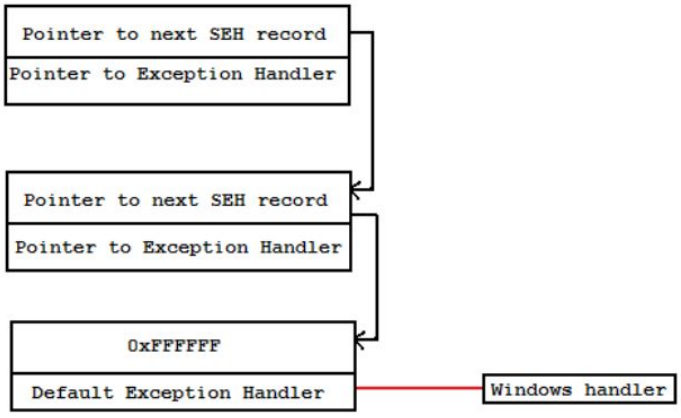
Canaries

Canary words are known values that are placed between a buffer and control data on the stack to monitor buffer overflows. When the buffer overflows, the first data to be corrupted will usually be the canary, and a failed verification of the canary data will therefore alert of an overflow, which can then be handled, for example, by invalidating the corrupted data.

Address space randomization

Address Space Layout Randomization (ASLR) is a memory-protection process for operating systems that guards against buffer-overflow attacks. It helps to ensure that the memory addresses associated with running processes on systems are not predictable, thus flaws or vulnerabilities associated with these processes will be more difficult to exploit.

ASLR increases the control-flow integrity of a system by making it more difficult for an attacker to execute a successful buffer-overflow attack by randomizing the offsets it uses in memory layouts.



Structured exception handling

Exceptions have complex structure, which defines how the exceptions are handled.
Exception handling is implemented as a list. From most detailed exception handler to default ex. handler.
It the handler cannot handle the ex. it passes it forward.

If the attacker would be able to replace code of the hadler for its own (for example via buffer overflow), that would not resend the exception to the next handler but instead causes the application to fall down..
Protection against this, is, that before the exception proceeds to handling, the windows handler validates, that the chain is complete, aka it ends in the default ex. handler.

Write / execute bit for memory pages
Mechanism to prevent trivial buffer overflow attack.
Each memory unit on the heap has bit of value either write OR execute.
Which means that only write or execute action may be performed on that block of memory.
The attacker than cannot write to the heap and then execute code which he wrote there.

In computer security, **executable-space protection** marks memory regions as non-executable, such that an attempt to execute machine code in these regions will cause an exception. It makes use of hardware features such as the NX bit (no-execute bit), or in some cases software emulation of those features. However technologies that somehow emulate or supply an NX bit will usually impose a measurable overhead; while using a hardware-supplied NX bit imposes no measurable overhead.
Executable bit is used in most of the today's systems: Android, Linux, Windows, ...

Lecture 6 – Running not so trusted code

Confinement mechanism

Confinement mechanism deals with prevention of a process to take disallowed actions.
If we want to use a software that we don't trust, we want to isolate is as much as possible.

We distinguish different levels of confinement

- Air-gaps
- Virtual machines
- Sandboxing (Containers)

Air-gap Isolation

Refers to a system, physically detached from network or other means of interaction with other systems.

Most important rules for creating Air-gap isolated systems:

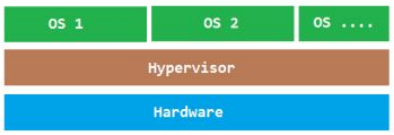
- During set-up use as little internet as possible.
- Turn on encryption.
- Install minimal software you need.
- Once set-up, never connect it to the internet.
- Install only software downloaded anonymously on different computer, check signatures and fingerprints
- Disable all autoruns.
- Minimize the amount of executable code moved to the computer (includes macros in text documents, pdfs).
- Use only trusted media. CDROM is more secure then USB stick.

Virtual machines

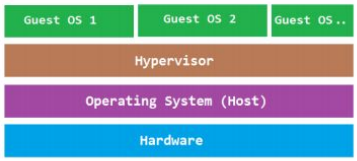
is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination.

Type I
hypervisor on top of HW

Type II
hypervisor on top of host OS



Type I

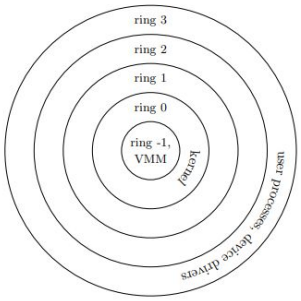


Type II

Security of VM is similar to security of OS. The crucial parts are separation of memory and proper system calls.

+ its possible to use multiple virtualized OS one one machine
+ we can have pre configured OS which we can run in virtual

- VM have overhead while used
- If hardware fails all the infrastructure may be lost

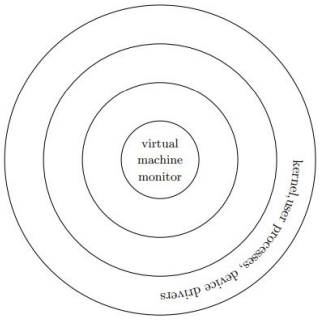


Isolation of CPU with HW support

Added new ring for VM with instructions supporting VMM / VM switch

VMM in ring -1

Isolation of CPU without HW support



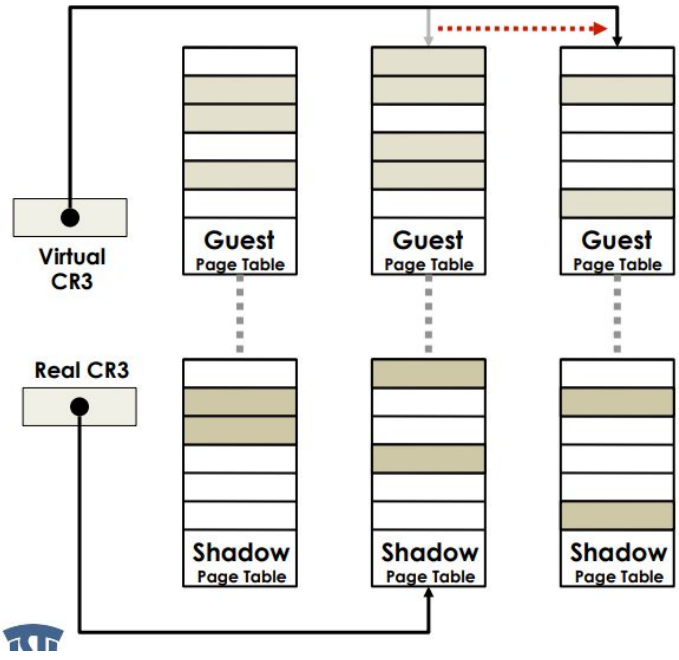
Paging in virtual machines

If the system does not have hardware support for virtualisation, the Virtual Machine Monitor, has to manage the paging of the virtualized OS.

Shadow paging

Shadow paging is used to map guest page tables to shadow page tables.

The **hypervisor intercepts instructions** which manipulate guest's memory and instead of pointing to the regular page table it changes the Control register, CR3 to point to the shadow table instead.



Shadow page tables are used by the hypervisor to keep track of the state in which the guest "thinks" its page tables should be. The guest can't be allowed access to the hardware page tables because then it would essentially have control of the machine. So, the hypervisor keeps the "real" mappings (guest virtual -> host physical) in the hardware when the relevant guest is executing, and keeps a representation of the page tables that the guest thinks it's using "in the shadows.

As far as page faults go, nothing changes from the *hardware's* point of view (remember, the hypervisor makes it so the page tables used by the hardware contain GVA->HPA mappings), a page fault will simply generate an exception and redirect to the appropriate exception handler. However, when a page fault occurs while a VM is running, this exception can be "forwarded" to the hypervisor, which can then handle it appropriately.

Nested paging

Nested paging uses an additional nested page table (nPT) to translate guest physical addresses to system physical addresses

The gPT maps guest linear addresses to guest physical addresses. Nested page tables (nPT) map guest physical addresses to system physical addresses.

Guest and nested page tables are set up by the guest and hypervisor respectively. When a guest attempts to reference memory using a linear address and nested paging is enabled, the page walker performs a 2-dimensional walk using the gPT and nPT to translate the guest linear address to system physical address.

Nested paging removes the overheads associated with shadow paging. **Unlike shadow paging, once the nested pages are populated, the hypervisor does not need to intercept and emulate guest's modification of gPT.**

General-purpose of sandboxes

The best way to ensure that the application won't do anything harmful is to lock it inside the separate box**.**

Types of sandboxing

- **A.** General purpose - general sandbox, can be used to solve multiple isolation problems
- **B.** Single purpose - specialized sandboxe**s**

In computer security, a "sandbox" is a security mechanism for separating running programs, usually in an effort to mitigate system failures

Sandboxing and Kernel

Kernel has its own security monitor. if we want to add our custom sec. monitor, we have two basic options where to put it.

A. Outside the kernel (to tho application ring)
    a. not used anymore, hard to achieve security
    b. probably because it's in user space, so it may be altered?
B. Inside the kernel itself - Our sec. monitor will wrap syscalls

Now we use third option - instead of implementing our own security monitor, we enlarge functionality of kernel sec. monitor.

Operating system level virtualization (Containerisation)

Example: Docker containers - saves the space by layering the OS

Docker: - initially not build for security usage, but for replication of environment, but still can be used to improve security

Mechanisms supporting general-purpose sandboxes on Linux (aka containers)

- **Kernel namespaces** - Namespaces provide the first and most straightforward form of isolation: processes running within a container cannot see, and even less affect, processes running in another container, or in the host system.

  Each docker obtains its own namespace. Process from one docker, cannot influence process of other docker, because it does not know his namespace.

- **seccom (+ bpf)** - Secure computing mode (seccomp) is a Linux kernel feature. You can use it to restrict the actions available within the container. The seccomp() system call operates on the seccomp state of the calling process. You can use this feature to restrict your application's access. Prohibits all syscalls except fork. The version **seccomp-bpf** is more flexible and allows us to write custom rules.
- **CGROUP** - (Control Groups) - umoznuje definovat restrictions na resources - pocet CPU, množství přidělené paměti apod.
  They provide many useful metrics, but they also help ensure that each container gets its fair share of memory, CPU, disk I/O; and, more importantly, that a single container cannot bring the system down by exhausting one of those resources.
- **LKC (Linux Kernel Capabilities)** - umoznuje udelat restrictions na porty, I/O, apod.

- SELinux - I think this was completely secure version of linux???

The biggest weakness in this system level virtualization is that the attacker is only one layer from kernel. And we he manages to escape to kernel, the whole system is his.

Lecture 7 – Anatomy of a browser

Foreword:

Browser basically interpret/execute the code which is Turing complete and users might not know what it will do. For example HTML5 + CSS is accidentally Turing complete language. Browsers consist among other components also from the extensions and plugins (described later). The problem with this model is that those who program the extensions and plugins usually don't know anything about the security. Current browsers are very complex, the Google browser is composed of 5 millions lines of code and is written in 35 languages. Moreover, modern browsers are implementing their virtual machines for example to interpret JavaScript. On such a huge scale, it is impossible to avoid bugs. Just to give you an idea about it sources, think about html. Every browser is interpreting html differently according to standard. But no one is able to follow standards precisely and therefore it is being unintentionally bended during implementation of interpreter. This is especially dangerous because can create an escape bugs. Moreover, browser contains many more other features or modules like 3D graphics, GPS position, etc, therefore it is very complex and dangerous also.

Why is security of Browser so important?

Browsers interpret unknown code, so it is hard to enforce security.
Extensions for browsers are usually written by people who know nothing about security.
Most of the people uses Chrome -> one bug, means one thread to all users.

Why it is hard to achieve security in browsers?
Browser has to contain VM for JavaScript interpretation.
Browser has usually access to many system components -> GPS, video player, 3D viewers, etc.

Difference between plugin and extension and impact on security

- **Extension –** runs inside the browser as a part/modification of the browser
    - usually just a source code
    - chrome/mozilla has thousands of extensions
- **Plugin –** not part of a browser, only runs in one of its windows, but is executed as separate process outside the browser.
    - always executable
    - as of 2019, plug-ins have been deprecated by most browsers
    - only one plugin in chrome -> Adobe Flash Player

Sandboxing inside browsers

- separate browser into **broker process** (browser) and **target process** (page, extension). **Broker process** is basically the **reference monitor.** Broker process is monitoring accesses of other processes and is enforcing our policy.
- broker has user privileges, target has minimal necessary privileges
- **Chrome**
    - each page runs in separate process same as every extension
    - Firefox also adopted this procedure
    - example of processes: file system process, networking process,...

Mechanism on Linux

a) Each process has his own set of rights/privileges.
b) Each process can obtain its own **namespace**. Result: Rendering core process for example cannot access the file system process, because it does not know his namespace.
c) **Limiting syscalls** only to those which are needed.

Mechanism on Windows

a) **Security descriptor** – přiřazen každému objektu (Similarly, to Linux where everything is an object, on Windows, everything is an object.), zajišťuje discretionary access control, definuje co může descriptor objekt dělat a co ne.

By applying these techniques we apply **Least Privilege Principle**.

Architecture of extensions

Privilege separation

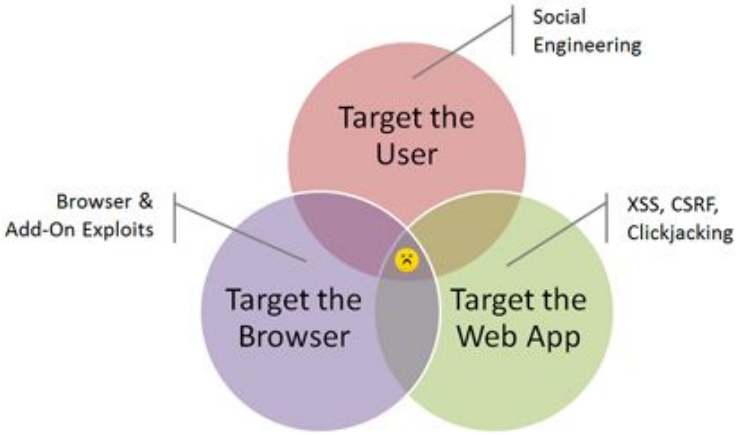
- in Firefox extensions originally ran with same privilege as browser
- in Chrome extensions ran with privileges described in manifest.json
    - to prevent developers using the maximum privileges, the Google uses lengthy procedure for allowing extensions with such privileges
    - it also contains web site access, API access, Execute Arbitrary Code
- the extension is separated into three components, between each two components is a process boundary:
    - **Content script**
        - has direct access to the DOM of a single web page and is thereby exposed to potentially malicious input. However, content scripts have no other privileges except for the ability to send messages to the extension core.
    - **Extension core**
        - contains the bulk of the extension privileges, but the extension core can only interact with web content via `XMLHttpRequest` and content scripts. Even the extension core does not have direct access to the host machine.
    - **Native Binary**
        - optional
            - requires maximum privileges
        - can access the host machine with the user's full privileges. The native binary interacts with the extension core via the standard NPAPI interface used by Flash and other browser plug-ins

Isolation mechanism

- The content script runs in an isolated world. Instead of accessing the underlying DOM data structures via the same JavaScript objects used by the page, each content script accesses the DOM with its "own" JavaScript objects. Content scripts and web pages therefore never exchange JavaScript pointers, making it more difficult for a malicious web page to confuse the content script (e.g.,with a JavaScript rootkit).

Attack vectors

An attack vector is a path or means by which a hacker can gain access to a computer or network server in order to deliver a payload or malicious outcome. Attack vectors enable hackers to exploit system vulnerabilities, including the human element.



Three web attack vectors seem to be responsible for the majority of computer attacks that involve a web browser:

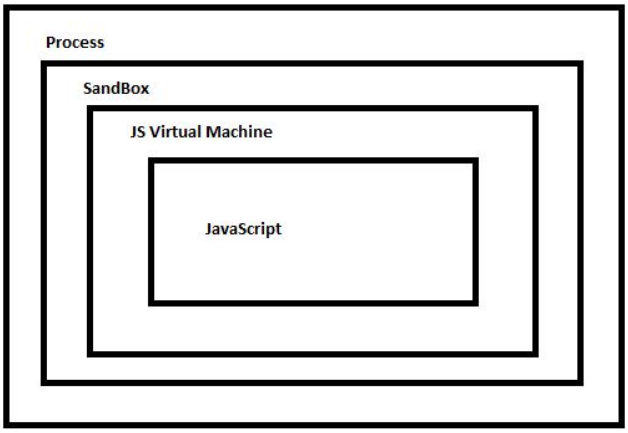
- The attack can incorporate an element of **social engineering** to persuade the victim to take an action that compromises security. For instance, the victim can supply data to a phishing site or install a program that will turn out to be malicious.

- The attacker can use the **browser as a gateway** for attacking web applications via techniques such as **cross-site scripting** (XSS), **Cross-Site Request Forgery** (CSRF) and **Clickjacking**.

- The attacker can **exploit a vulnerability in the web browse**r or in local software that the browser can invoke. Such client-side exploits have targeted browser add-ons such as Flash, Adobe Reader and Java Runtime Environment (JRE).

JavaScript Virtual Machine

A JavaScript engine is a computer program that executes JavaScript (JS) code. The first JavaScript engines were mere interpreters, but all relevant modern engines utilize just-in-time compilation for improved performance.



JavaScript engines are typically developed by web browser vendors, and every major browser has one. The use of JavaScript engines is not limited to browsers. For example, the Chrome V8 engine is a core component of the popular Node.js runtime system.

How malware can escape from JVM

How is it possible that the malicious code gets from JS Virtual machine to the host computer?

JSVM has its own virtual memory.
Sandbox has its own virtual memory.

But from the Kernel point of view, it is one process, which obtains one/more pages from the VM. For the malicious JS the only necessary thing to do is to allocate huge string/array/… which exceeds the size of JSVM virtual memory. Then it can start to modify the VM sandbox.

Remote browsing / browser isolation

The core concept behind browser isolation is security-through-physical-isolation to create a "gap" between a user's web browser and the endpoint device thereby protecting the device (and the enterprise network) from exploits and attacks. Unlike secure web gateways, antivirus software, or firewalls which rely on known threat patterns or signatures, this is a zero-trust approach.

There are two primary browser isolation architectures:
(1) client-based *local* isolation and (2) *remote* isolation.

Local browser **isolation** attempts to isolate a browser running on a local endpoint using app-level or OS-level sandboxing. In addition to leaving the endpoint at risk when there is an isolation failure, these systems require significant endpoint resources (memory + compute), tend to be brittle, and are difficult for IT to manage as they depend on support from specific hardware and software components.

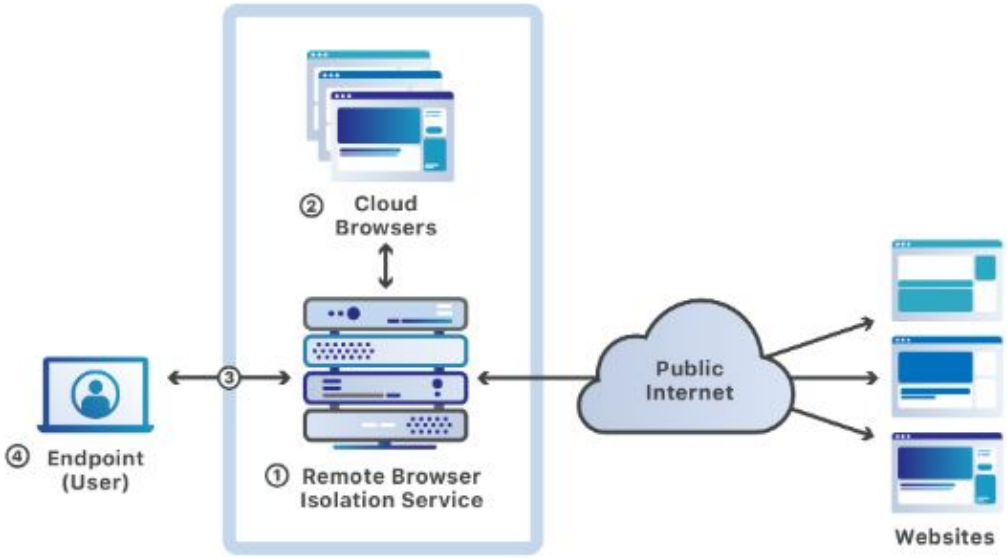
Remote browser isolation (RBI) protects the endpoint by moving the browser to a remote service in the cloud or to a separate on-premises server within the enterprise network:

- On-premises isolation simply relocates the risk from the endpoint to another location within the enterprise without actually eliminating the risk.
- Cloud-based remote browsing isolates the end-user device and the enterprise's network while fully enabling IT control and compliance solutions.

How does Remote Browser Isolation (RBI) work?

In a typical cloud-based RBI system (the blue-dashed box ❶ below), individual remote browsers ❷ are run in the cloud as disposable containerized instances – typically, one instance per user. The remote browser sends the rendered contents of a web page to the user endpoint device ❹ using a specific protocol and data format ❸. Actions by the user, such as keystrokes, mouse and scroll commands, are sent back to the isolation service

over a secure encrypted channel where they are processed by the remote browser and any



resulting changes to the remote browser webpage are sent back to the endpoint device.

In effect, the endpoint device is “remote controlling” the cloud browser.

Lecture 8 - Access control of web (Tangled web)

Threat landscape of browsers and web

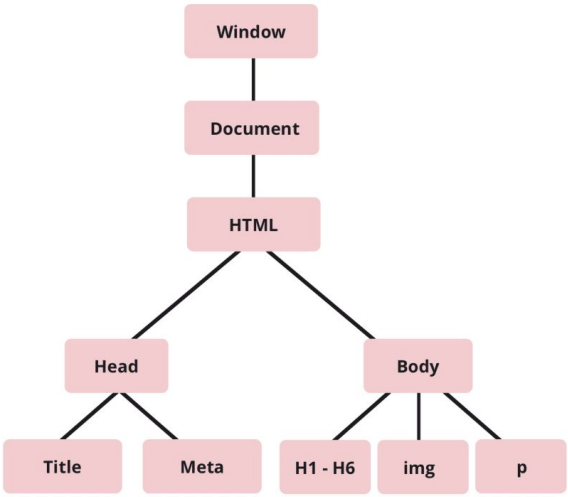
The problem of the web is that there are no exact boundaries between code and the data. Therefore R/W mechanisms don't work at the web. When downloading things from the web, we have no control of what will execute and we cannot check it beforehand. Web is a client-server model, known from 60'th. Client server model, can be made reasonably secure, but! not in the case of web. Imagine that we have a web blog where anyone can insert content. This means that we have untrusted data, this leads to cross site scripting. Browser can't distinguish, which data are trusted and which are untrusted.

Regarding the client server model, the client has no way to distinguish between trusted and untrusted data. All data send by the server are executed automatically. The same applies to server side. The difference between the traditional client-server model is, that in the web, the client connects, does something and disconnects. After it disconnects the CS model forgets everything. Web has to use **cookies** to maintain at least some integrity of communication.

Web pages consist of large number of components, from different sources. Unlike for example programs (Word, VLC player etc. which are complete units created by one source.). Therefore on the web is hard to determine which components are safe and which are not. Also it is hard to determine how these components should interact with each other. For this a **same origin policy** was defined.

Same-origin policy

Each resource has defined origin and can only access resources from its origin. Simple motto: TRUST THE DATA FROM THE SAME ORIGIN, UNTRUST THE OTHERS.
Origin = scheme + hostname + port
(ex. http://food.com/index.html (port is implicitly 80))



Under the policy, a web browser **permits** scripts contained in a first web page to access data in a second web page, but **only if** both web pages have **the same origin**.

This policy prevents a malicious script on one page from obtaining access to sensitive data on another web page through that page's Document Object Model.

Most of the checks are done on the client side, server does not check the origin.

Four basic ideas:

- each origin has client side resources (ex: cookies, DOM storage, JS namespace. DOM tree)
- Each frame gets the origin of its URL
- Scripts execute with the authority of its frame's origin
- Passive content gets zero authority (ex: images, css)

Two components communicate via Post Messages.

Frame - component of DOM object model, used for JS.
Frames/ windows objects get origin of the frame's URL **OR** a suffix of the original domain
Frame may change its document.domain name to suffix of its doc.domain name.
(ex. x.y.z.com -> y.z.com (in order to communicate with other frame)), but! ->

Two frames can interact with each other, if and only if a) or b) is true.

- a) Both frames set its document.domain to the same value
- b) Neither has changes its document.domain value.

The following table gives an overview of typical outcomes for checks against the URL "**http://www.example.com/dir/page.html**".

Compared URL	Outcome	Reason
http://www.example.com/dir/page2.html	Success	Same scheme, host and port
http://www.example.com/dir2/other.html	Success	Same scheme, host and port
http://username:password@**www.example.com**/dir2/other.html	Success	Same scheme, host and port
http://www.example.com:**81**/dir/other.html	Failure	Same scheme and host but different port
https://www.example.com/dir/other.html	Failure	Different scheme
http://**en.example.com**/dir/other.html	Failure	Different host
http://**example.com**/dir/other.html	Failure	Different host (exact match required)
http://**v2.www.example.com**/dir/other.html	Failure	Different host (exact match required)

Cookies

Cookies has are defined by domain + path ->(ex. .mit.edu/9.125)
They also may have a secure flag (which means that they belong to https and http cannot use it). Cookies have different same-origin policy. From cookie perspective the root domain is the same origin, therefore if we set cookie to cvut.cz/pubs then even kos.cvut.cz/index has access. Cookies are dangerous and used for **cross-site request attack.**

CSS, Images, Javascript

CSS and images are treated in a similar way, but CSS is Turing complete language! Javascript is interpreted in own virtual machine in the browser. More on that in previous chapter.

Mashups

A mashup (computer industry jargon), in web development, is a web page or web application that uses content from more than one source to create a single new service displayed in a single graphical interface. For example, a user could combine the addresses and photographs of their library branches with a Google map to create a map mashup.

Tzn. Každá dnešní stránka je mashup.

HTML permissive parsing

The `html-parsing` parsing behavior is permissive in that it accepts erroneous HTML, handling several classes of HTML syntax errors gracefully, without yielding a parse error. -> Tzn. parser je schopen detekovat chyby v syntaxi a opravit je, coz ale znamená utocnik muze teto vlastnosti vyuzit k útoku.

Př: html parser se snaží opravit html stánku, která obsahuje spustitelný exploit and voilá, útok je tu :D

Sandboxing with HTML-5

In short: If we want to place 3rd-party content on our web-site (advertisements, blog comments, widgets, etc.) we put our self at risk of attack such as cross-site scripting (XSS), phishing, or information disclosure from this content. HTML-5 provides a new way of dealing with such content, known as iframe.

By placing the content in an **iframe**, the developer can specify the sandbox attribute on the iframe to apply a set of basic security restrictions:

- Plugins are disabled.
- Script execution is blocked.
- Form submission is blocked.
- The content is treated as if it was from a globally unique origin. Meaning, all APIs which require same-origin (such as localStorage, XMLHttpRequest, and access to the DOM of other documents) are blocked.
- The content is blocked from navigating the top level window or other frames on the page (excluding child frames of the sandboxed content).
- Popup windows are blocked.

Content security policy

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks.

Configuring Content Security Policy involves adding the Content-Security-Policy HTTP header to a web page and giving it values to control resources the user agent is allowed to load for that page. For example, a page that uploads and displays images could allow images from anywhere, but restrict a form action to a specific endpoint. A properly designed Content Security Policy helps protect a page against a cross site scripting attack.

Server administrators can specify the domains that the browser should consider to be valid sources of executable scripts.

HTTP Strict Transport Security (HTTP STS)

The HSTS Policy is communicated by the server to the user agent via an HTTPS response header field named "Strict-Transport-Security". HSTS Policy specifies a period of time during which the user agent should only access the server in a secure fashion. **Websites using HSTS often do not accept clear text HTTP, either by rejecting connections over HTTP or systematically redirecting users to HTTPS** (though this is not required by the specification). The consequence of this is that a user-agent not capable of doing TLS will not be able to connect to the site.

Cross-site-scripting

XSS enables attackers to inject malicious scripts into web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same-origin policy. Cross-site scripting carried out on websites accounted for roughly 84% of all security vulnerabilities documented by Symantec up until 2007.

Cross-site-request forgery

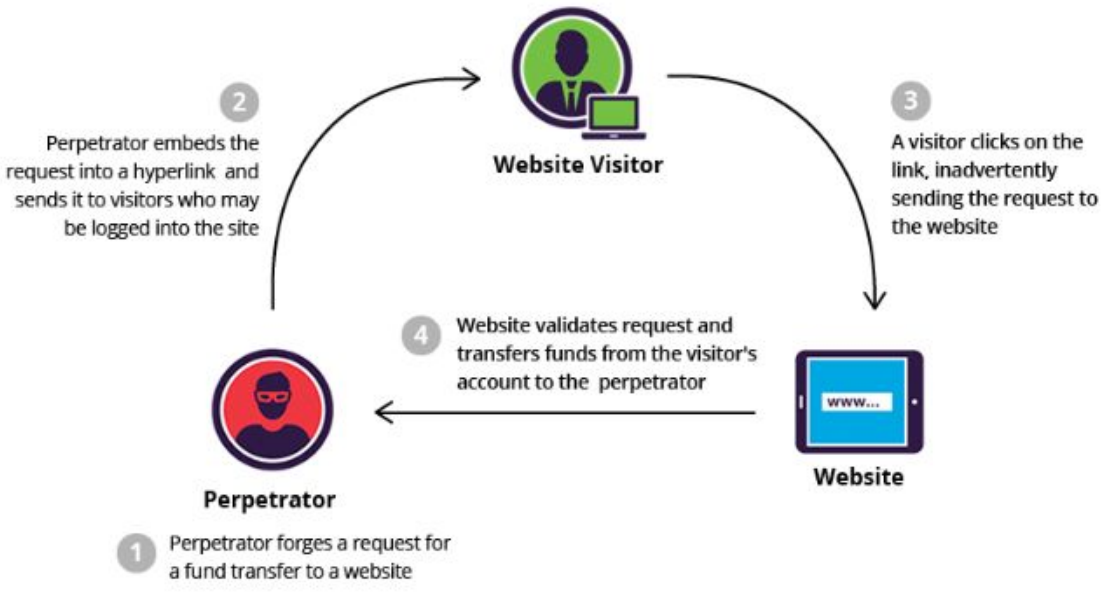
MIT example: back.com/transferAmount=500$ & to = atackerAcountNumber

The user logs to the bank, the clicks on button of the attacker which send the prepared request above. So the attacker uses, the user who logs to his account and then just send the funds from victim's account.

Cross site request forgery (CSRF), also known as XSRF, Sea Surf or Session Riding, is an attack vector that tricks a web browser into executing an unwanted action in an application to which a user is logged in.

A successful CSRF attack can be devastating for both the business and user. It can result in damaged client relationships, unauthorized fund transfers, changed passwords and data theft—including stolen session cookies.

CSRFs are typically conducted using malicious social engineering, such as an email or link that tricks the victim into sending a forged request to a server. As the unsuspecting user is authenticated by their



application at the time of the attack, it's impossible to distinguish a legitimate request from a forged one.

Lecture 9 - Legacy protocols

Foreword:

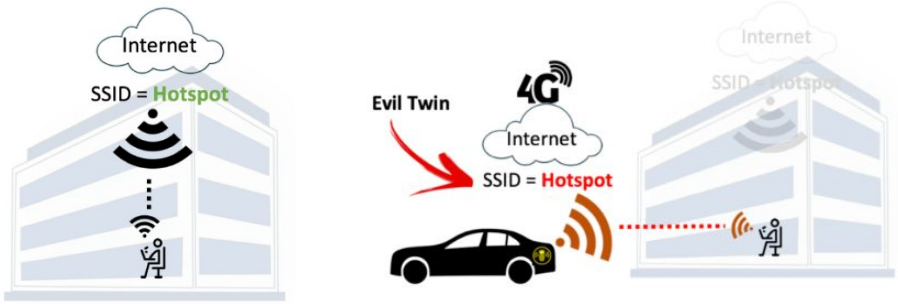
This lecture is about security problems in protocols.

The problem with legacy protocols (first invented protocols) is that they were designed to be functional not secure. No one assumed that these protocols could be used to attack computer networks.

We distinguish two basic types of protocol attacks.

- **Man in the middle** – the attacker secretly relays and possibly alters the communications between two parties who believe that they are directly communicating with each other.
- **Eavesdropping** – secretly listening to the communication - for example MAC flooding

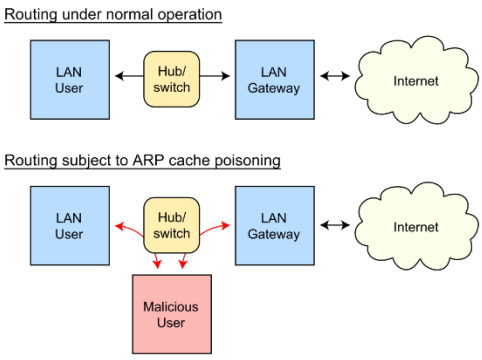
Evil twin attack



The principle of evil twin attack is fairly simple a threat actor broadcasts the same SSID as the legitimate AP (and often the same BSSID or MAC address of the SSID) to fool the device into connecting. Once the user connects to the internet via evil twin. The whole communication passes through the attackers machine. Which means that he can observe the traffic. Eardrop the credentials etc.

ARP spoofing

In computer networking, ARP spoofing, ARP cache poisoning, or ARP poison routing, is a technique by which an attacker sends (spoofed) Address Resolution Protocol (ARP) messages onto a local area network. Generally, the aim is to **associate the attacker's MAC address with the IP address of another host**, such as the default gateway, causing any traffic meant for that IP address to be sent to the attacker instead.
Note: No easy solution available because the problem is inside the protocol itself. Modification of the protocol is impossible since its used everywhere.



IP protocol

Weaknesses of IP protocol:

- No encryption – everything is visible
- No integrity checks – the only check is for the random bit change during the transfer
- Possible threats: Internet Control Message Protocol (ICMP) echo packets (pings) to discover subnets and hosts on a protected network, to generate DoS flood attacks, and to alter host routing tables. IP spoofing, etc.
- No guarantee it will arrive. Goal of IP is routing through the internet -> kinda tries to deliver the message.
- Has time-to-live so it won't circulate net forever.
- Possible attack - **IP spoofing**. - podvržení zdrojové adresy paketu - tj pakety putují na jinou ip adresu, IP spoofing lze použít k DOS attacku
- Obrana: Odchycení paketů z vnější sítě, které se tváří jako pakety mezi dvěma počítači z vnitřní sítě.

TCP

- Reliable
- Full duplex – every send packet contains sequence number, after successful transfer the number is increased, if the number does not match -> abort. The attacker cannot interfere, if he doesn't know the number.
- **The number is increased by length of the message**. The starting number is a random value generated as: **Hash (IP, DST, IP, SOURCE, salt)**. This way parties are somehow enforcing the integrity, because they can check the value was properly increased, but the integrity stands on one number only. It relies on that this number can't be guessed by anyone. It is mainly designed to protect from interfering connections but not from the attackers.

- A TCP connection is established in three steps:
  - The initiating client requests a client-to-server communication session with the server. SYN -> SYN
  - The server acknowledges the client-to-server communication session and requests a server-to-client communication session. SYN, ACK -> SYN, ACK
  - The initiating client acknowledges the server-to-client communication session. ACK -> ACK
- TCP SYN Flood attack
  - exploits the TCP three-way handshake. The threat actor continually sends TCP SYN session request packets with a randomly spoofed source IP address to an intended target. The target device replies with a TCP SYN-ACK packet to the spoofed IP address and waits for a TCP ACK packet. Those responses never arrive. Eventually the target host is overwhelmed with half-open TCP connections and denies TCP services to legitimate users.

- TCP reset attack
  - Can be used to terminate a TCP communication between two hosts. A threat actor could send a spoofed packet containing a TCP RST to one or both endpoints. This would cause them to immediately stop communicating.

- TCP session hijacking

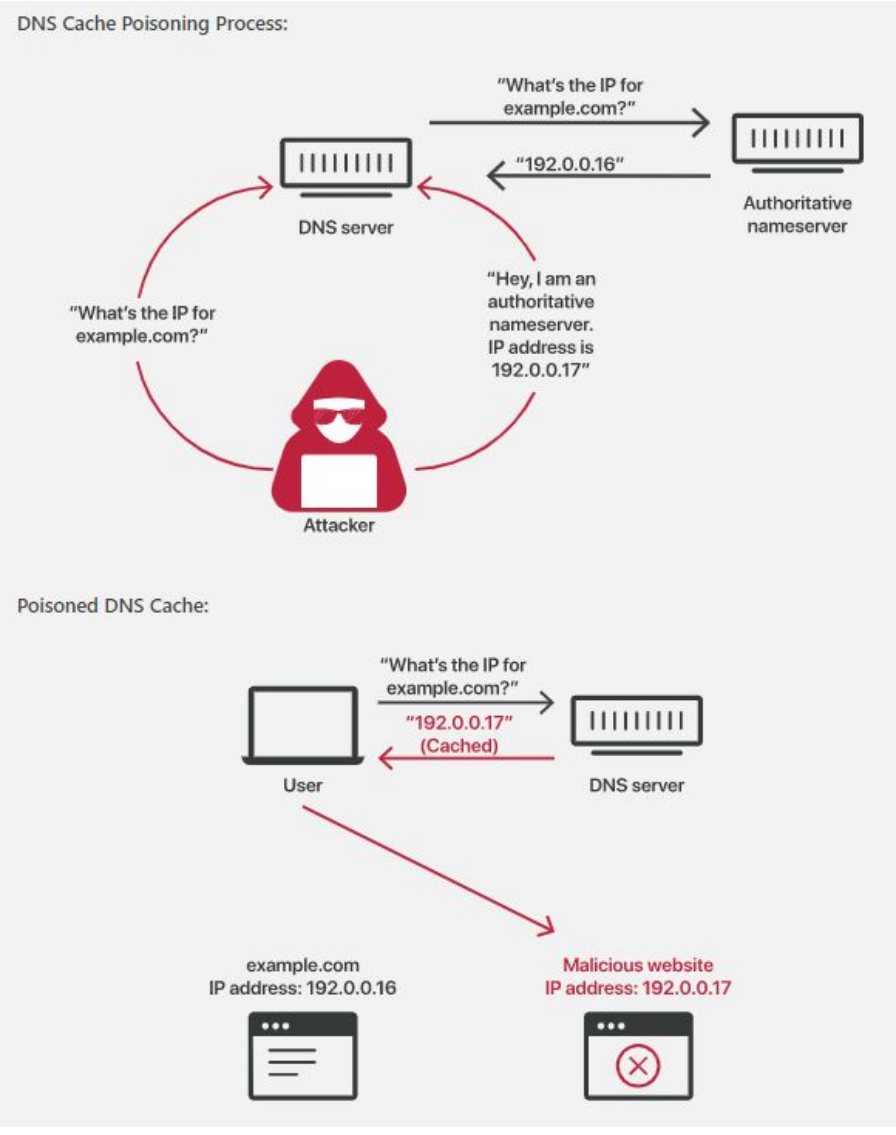
  - **TCP/IP Hijacking** is when an unauthorized user hijacks a network connection of another user.
  - For example, the attacker monitors the network transmission and analyzes the source and destination IP addresses of the two computers.
  - Once the attacker discovers the IP address of one of the users, the attacker can knock one of the users off their connection using a denial of service (DoS) attack or other type of attack and then resume communication by spoofing the IP address of the disconnected user.
  - The other user is tricked into thinking he is still talking to the same legitimate user.

- The Mitnick's attack - http://wiki.cas.mcmaster.ca/index.php/The_Mitnick_attack. As mentioned above, the communication is protected by communicating one number starting from random value and increasing by the length of the message. Unfortunately, the standard defines, that the number should be increased roughly by 250000/s and that makes it guessable. First such an attack was done by Kevin Mitnick. It was solved by adding salt (random trash) to the hash. And the salt is different for each connection.

UDP and DNS

Imagine that, as a senior-year prank, high school seniors change out all the room numbers on their high school campus, so that the new students who don't know the campus layout yet will spend the next day getting lost and showing up in the wrong classrooms. Now imagine that the mismatched room numbers get recorded in a campus directory, and students keep heading to the wrong rooms until someone finally notices and corrects the directory.

DNS cache poisoning is the act of entering false information into a DNS cache, so that DNS queries return an incorrect response and users are directed to the wrong websites. DNS cache poisoning is also known as 'DNS spoofing.' IP addresses are the 'room numbers' of the Internet, enabling web traffic to arrive in the right places. DNS resolver caches are the 'campus directory,' and when they store faulty information, traffic goes to the wrong places until the cached information is corrected. (Note that this does not actually disconnect the real websites from their real IP addresses.)



Because there is typically no way for DNS resolvers to verify the data in their caches, incorrect DNS information remains in the cache until the time to live (TTL) expires, or until it is removed manually. A number of vulnerabilities make DNS poisoning possible, but the chief problem is that DNS was built for a much smaller Internet and based on a principle of trust (much like BGP).

BGP (Border Gateway protocol)

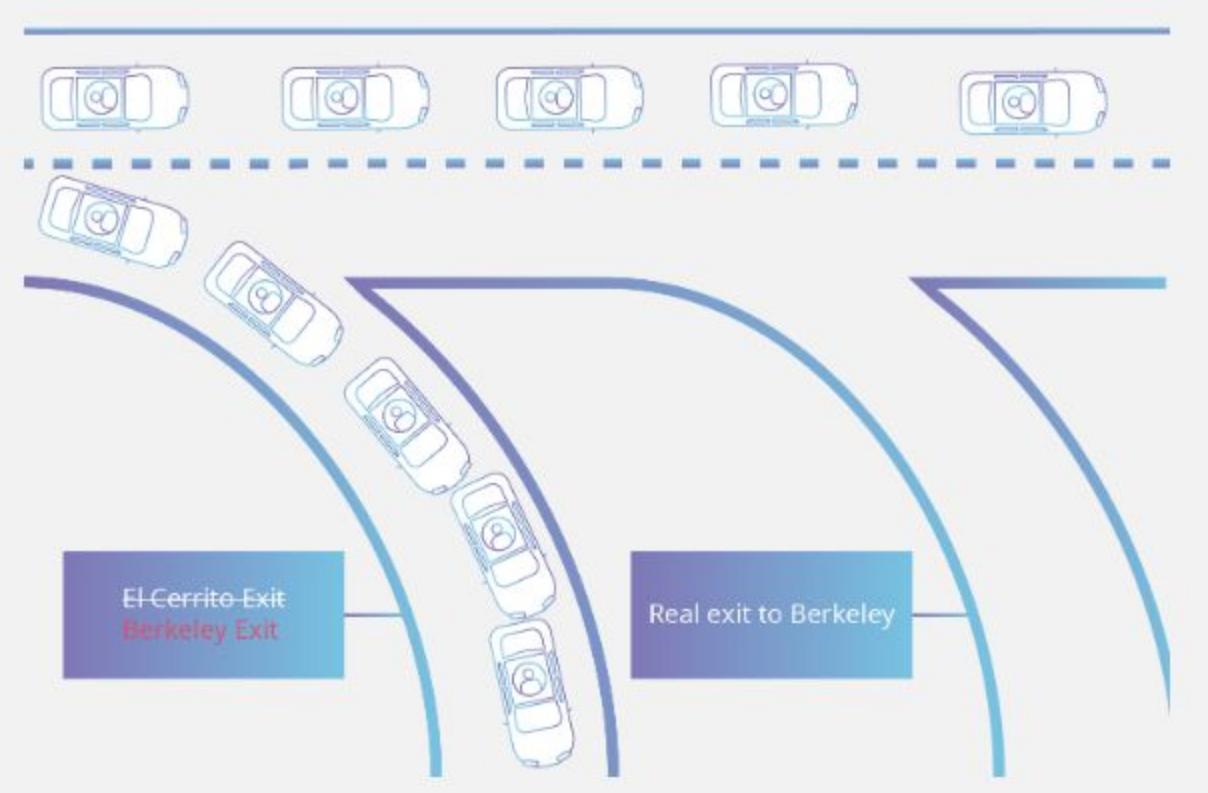
How it works and MITM https://dyn.com/wp-content/uploads/2013/05/blackhat-09.pdf

BGP - used for routing two autonomous systems. V síti existuje mnoho cest, kudy mohou putovat pakety mezi dvěma body.

Used for distributing of route tables of IP addresses. This protocol does not have any encryption nor integrity checks. Protocol updates route tables in routers. It obeys two rules:

1) Shorter path wins
2) More specific path/subnet wins

BGP hijacking is when attackers maliciously reroute Internet traffic. Attackers accomplish this by falsely announcing ownership of groups of IP addresses, called IP prefixes, that they do not actually own, control, or route to. A BGP hijack is much like if someone were to change out all the signs on a stretch of freeway and reroute automobile traffic onto incorrect exits



How can BGP be hijacked?

When an AS announces a route to IP prefixes that it does not actually control, this announcement, if not filtered, can spread and be added to routing tables in BGP routers across the Internet. From then until somebody notices and corrects the routes, traffic to those IPs will be routed to that AS. It would be like claiming territory if there were no local government to verify and enforce property deeds.

Specifičnost záznamu je určena množstvím známých pozic v ip adrese.

Least specific: 0.0.0.0/0

More specific: 1.0.0.0/0

More specific. 1.0.1.0/0

More specific: 1.0.1.0./25

Lecture 10 – Secure Alternatives

Key exchange with public-private keys

HTTPs communication:

Client <----------(Confidentiality, Integrity) ----------> Server

How to achieve confidentiality? -> Encryption
For encryption, client and server needs to exchange keys.
How to implement key exchange?

Version 1) Have a key distribution center
Problem: One ultimate key distributor is too much power -> trust issue and also a bottle neck
2) Key exchange without KDC

Key exchange without KDC:

- Server provides a public key.
- Client generates his public key, which is encrypted via public server key.
- Only server is able to decrypt clients message with his private key.
- Now only the server has clients public key -> aka connection established.

How to verify that the public key of the server is actually public key of our desired server?
Public key can be verified from 3rd party authority.

Signing the key
Server:
Text(verification) -> hash() -> hashed text -> encrypted via private key of the authority -> signature
Client:
Set of public keys of certified authorities stored in our PC -> public key -> hashed text->unhash() ->text
Client has a set of public keys of certified authorities -> aka is able to decode the signature.

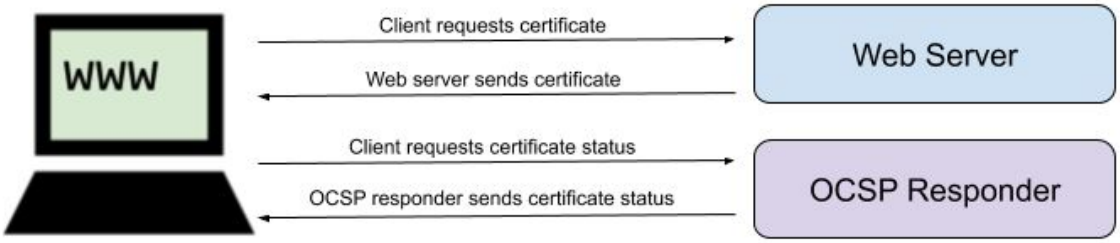
Problems of certificates --- a revocation nightmare

What to do if we detect an attacker?

- Put the attacker on the black list -> list of untrusted certificate agencies (CA)
- Remove certificate agency from the list of secure agencies
- OSCP - The Online Certificate Status Protocol (OCSP) is the Internet protocol used by web browsers to determine the revocation status of SSL/TLS certificates supplied by HTTPS websites. While SSL/TLS certificates are always issued with an expiration date, there are certain circumstances in which a certificate must be revoked before it expires (for example, if its associated private key may have been compromised). Therefore, the current validity of a website's certificate must always be checked by clients regardless of its expiry date.

In its simplest form, OCSP works as follows:

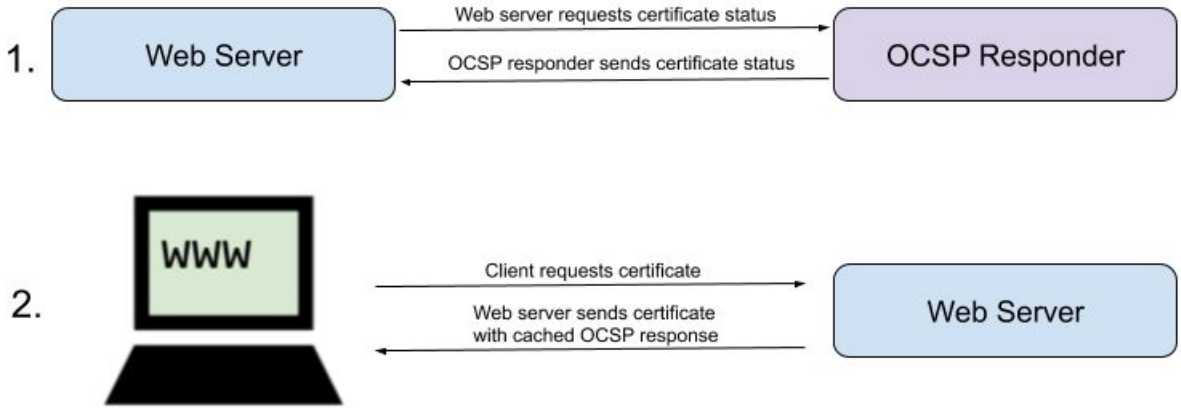
1. A web browser receives a certificate from an HTTPS website.
2. The web browser sends a request to an OCSP responder, a server operated by the certificate authority (CA) that issued the certificate.
3. The OCSP responder's signed response to the browser indicates whether the certificate is valid or has been revoked.



OSCP Stapling

While initially introduced to solve the bandwidth and scaling problems of **certificate revocation lists** (CRLs), OCSP introduced several performance and security issues of its own that are currently being addressed through **OCSP stapling**. In OCSP stapling:

1. A web server requests and obtains a signed OCSP response for its certificate from an OCSP responder, which can be cached for up to 7 days.
2. The server includes the cached OCSP response along with (or "stapled to") its certificate in its HTTPS responses to web browsers.
3. To prevent a potential attack in which a website serves a stolen revoked certificate without a stapled OCSP response, certificates may be issued with a **must-staple** extension, mandating OCSP stapling for the certificate.



OSCP - pomalejsi metoda overeni public key serveru, browser si sam overuje u oscp responderu zda je public key valid

OSCP STAPLING -rychlejsi verze, server si sam necha overit svuj klic (dostane watermarked overeni s oscp s omezenou dobou platnosti od oscp responderu) a ten posle klientovi

Attacks

Cache poisoning

Cache poisoning also known as DNS cache poisoning is described in previous lecture

HTTPS stripping (SSL stripping)

SSL/TLS is a secure protocol used to communicate sensitive information. This protocol is used when exchanging sensitive data such as banking information and email correspondence for example. The protocol's security is established by creating an encrypted connection between two parties (usually a client application and a server). Browsers and web servers regularly use this protocol when a secure connection is needed. In most scenarios the following events take place when establishing a secure connection:

1. The user sends unsecured HTTP request
2. The server answers via HTTP and redirects the user to secure protocol(HTTPS)
3. The user sends secure HTTPS request

How the ssl strip works?

In order to "strip" the SSL, an attacker intervenes in the redirection of the HTTP to the secure HTTPS protocol and intercepts a request from the user to the server. The attacker will then continue to establish an HTTPS connection between himself and the server, and an unsecured HTTP connection with the user, acting as a "bridge" between them.

SSL Strip attacks can be implemented in a number of ways. The most common method is by creating a hotspot and allowing the victims to connect to it. Many attackers establish fake hotspots with names similar to legitimate hotspot names, for example, "Starbucks Coffee" instead of "Starbucks". Unaware, the user connects to the malicious hotspot. Once the user tries to connect to the server, the attacker uses his control over the hotspot and attacks the user.

Protocol downgrading attack
A downgrade attack or version rollback attack is a form of cryptographic attack on a computer system or communications protocol that makes it abandon a high-quality mode of operation (e.g. an encrypted connection) in favor of an older, lower-quality mode of operation (e.g. cleartext) that is typically provided for backward compatibility with older systems.

HSTS
HTTP Strict Transport Security (HSTS) je v informatice bezpečnostní mechanismus, který chrání síťovou komunikaci mezi webovým prohlížečem a webovým serverem před downgrade útoky a zjednodušuje ochranu proti únosu spojení (tzv. cookie hijacking). Mechanismus umožňuje, aby webový server vynutil v prohlížeči komunikaci pouze pomocí šifrovaného HTTPS připojení a vyloučil tím přenos dat nezabezpečeným HTTP protokolem.

Certificate pinning
Typically certificates are validated by checking the signature hierarchy; `MyCert` is signed by `IntermediateCert` which is signed by `RootCert`, and RootCert is listed in my computer's "certificates to trust" store.

Certificate Pinning was where you ignore that whole thing, and say trust *this certificate only* or perhaps trust only certificates *signed by this certificate*, ignoring all the other root CAs that could otherwise be trust anchors. It was frequently also known as Key Pinning, since it was actually the public key hash that got saved.

But in practice, Key Pinning turned out to cause more problems than it solved. It was frequently misconfigured by site owners, plus in the event of a site compromise, attackers could maliciously pin a cert that the site owner didn't control. Key Pinning was deprecated in 2017, and was removed entirely from Chrome and Firefox in Nov. 2019. It was never supported to begin with by IE and Safari.

DNSSEC
The original design of the Domain Name System (DNS) did not include any security details; instead, it was designed to be a scalable distributed system. The Domain Name System Security Extensions (DNSSEC) attempts to add security, while maintaining backward compatibility. RFC 3833 documents some of the known threats to the DNS and how DNSSEC responds to those threats.

DNSSEC was designed to protect applications (and caching resolvers serving those applications) from using forged or manipulated DNS data, such as that created by DNS cache poisoning. All answers from DNSSEC protected zones are digitally signed. By checking the digital signature, a DNS resolver is able to check if the information is identical (i.e. unmodified and complete) to the information published by the zone owner and served on an authoritative DNS server.

HTTPS

In HTTPS, the communication protocol is encrypted using Transport Layer Security (TLS) or, formerly, its predecessor, Secure Sockets Layer (SSL). The protocol is therefore also often referred to as HTTP over TLS,or HTTP over SSL.

The principal motivations for HTTPS are authentication of the accessed website, protection of the privacy and integrity of the exchanged data while in transit. It protects against man-in-the-middle attacks. The bidirectional encryption of communications between a client and server protects against eavesdropping and tampering of the communication. In practice, this provides a reasonable assurance that one is communicating without interference by attackers with the website that one intended to communicate with, as opposed to an impostor.

Lecture 11 - Intrusion Detection and Prevention

Firewalls

- **Packet filtering (stateless) firewall**
  Packet filters act by inspecting packets transferred between computers. When a packet does not match the packet filter's set of filtering rules, the packet filter either drops (silently discards) the packet, or rejects the packet (discards it and generates an Internet Control Message Protocol notification for the sender) else it is allowed to pass. Packets may be filtered by source and destination network addresses, protocol, source and destination port numbers**.**
- **Circuit firewall**
  Controls if the packets comes from verified (established communication). Example: If the packet comes from port of established TCP communication, then it passes. Other packets won't pass.
- **Stateful firewall**
  Combination of packet and circuit firewall.
  In contrast a **stateless firewall** does not take context into account when determining whether to allow or block packets.
  This type of firewall is potentially vulnerable to denial-of-service attacks that bombard the firewall with fake connections in an attempt to overwhelm the firewall.
- **Application gateway firewall (proxy firewall)**
  The key benefit of application layer filtering is that it can understand certain applications and protocols (such as File Transfer Protocol (FTP), Domain Name System (DNS), or Hypertext Transfer Protocol (HTTP)). This is useful as it is able to detect if an unwanted application or service is attempting to bypass the firewall using a disallowed protocol on an allowed port, or detect if a protocol is being abused in any harmful way.

Firewalls are reference monitors

- unbypassable - nelze jej obejit
- tamper resistent - nelze jej prepsat
- verifiable

Intrusion deflection and honeypots

In computer terminology, a honeypot is a computer security mechanism set to detect, deflect, or, in some manner counteract attempts at unauthorized use of information systems. Generally, a honeypot consists of data that appears to be a legitimate part of the site, but is actually isolated and monitored, and that seems to contain information or a resource of value to attackers, who are then blocked. This is similar to police sting operations, colloquially known as **"baiting" a suspect.**

Intrusion detection systems

Designed to protect hosts against known and unknown malware. A IDS can perform detailed monitoring and reporting on the system configuration and application activity. It can provide log analysis, event correlation, integrity checking, policy enforcement, rootkit detection, and alerting. In order to work properly it has to be installed on every machine in the company, which is not always easy to achieve.

Techniques of intrusion detection

- signature matching
- behavior matching
- anomaly detection

Signature matching

Principle: We check packets for known attacks. If we know what packets were used in the previous attack. We can create hash/signature of that attack. The next time the attack happens, the packets are checked against the known database of malicious signatures.

Example: SNORT rules which determines if one or sequence of packets form known malicious pattern.

Pros. I

- Conceptually fairly simple
- Takes care of known attacks
- Easy to share signatures, build up libraries
- Can detect variants of known attacks

Cons.

- Size of the database (3500).
- Most time spent on signature matching.
- Cannot detect new threats or variants of existing threats.

Behavior matching

Principle: Behavior matching does not care about signatures, but instead it verifies the behavior of the users.

Attackers follow usually known patterns during the attack. For example, scanning the system for machines awake, scanning for open ports, etc. Behavior matching system tries to detect this behavior.

In the signature method, the system was pretty sure what is and what is not malicious activity. If the signature did not match, it was. But here, it is problematic to correctly define what is and what is not malicious. Which means that behavior matching has high false positive rate.

Anomaly intrusion detection

Principle: Detect anomalies in the system traffic. Anomaly detected when attack changes statistic distribution in the network.
Example: Brute forcing passwords -> network contains statistically important number of short connections.
Advantage -> anomaly detection is able to spot unknown attack types.
From the attackers point of view its best to hide the data into communication which is normal in the traffic. Therefore no statistically important increase of messages.

Lecture 12 - Denial of Service

DoS Attack – Type of an attack that does not steal data, but it aims to stop or force victim to stop providing given services.

Economic incentives of DOS

Purpose of DNS attack?

- Competition – example: florists on the valentine's day, games – motivate gamers to change the game
- No other legal option – example: taking down child pornography web page by denial of service
- Cyberwarfare - 'Cyberwarfare' is used in a broad context to denote interstate use of technological force within computer networks in which information is stored, shared or communicated online

- Its cheap to buy bot network to perform DOS attack, but results of such attack can be significant.

Denial of service at individual levels of OSI model

1. Fyzická vrstva – cutting wires, Wi-Fi radio jamming
2. Datová vrstva – MAC address flooding
3. Síťová vrstva – ICMP flooding, Smurf attack
4. Transportní vrstva – SYNC flood
5. Relační vrstva – Telnet exploit
6. Prezentační vrstva – Malformed SSL
7. Aplikační vrstva – HTTP GET/POST attack

DDos - Distributed DOS. Using multiple computers (often not yours but infected) it's possible to create attack taking bandwidth with higher efficiency and with lesser chance to be detected.

Cache poisoning - Corrupting DNS server and replacing legitimate adressis with spoofed ones

MAC Flooding

is a technique employed to compromise the security of network switches.

Switches maintain a MAC table that maps individual MAC addresses of the network to the physical ports on the switch. This allows the switch to direct data out of the physical port where the recipient is located, as opposed to indiscriminately broadcasting the data out of all ports as an Ethernet hub does. The advantage of this method is that data is bridged exclusively to the network segment containing the computer that the data is specifically destined for.

In a typical MAC flooding attack, a switch is fed many Ethernet frames, each containing different source MAC addresses, by the attacker. The intention is to consume the limited memory set aside in the switch to store the MAC address table.

The effect of this attack may vary across implementations, however the desired effect (by the attacker) is to force legitimate MAC addresses out of the MAC address table, causing significant quantities of incoming frames to be flooded out on all ports. Switch then works as hub, which means that everything it gets on input is broadcasted on all ports.

Defence: Subset of “known” addresses which is not rewritten. Or authentication with AAA server.

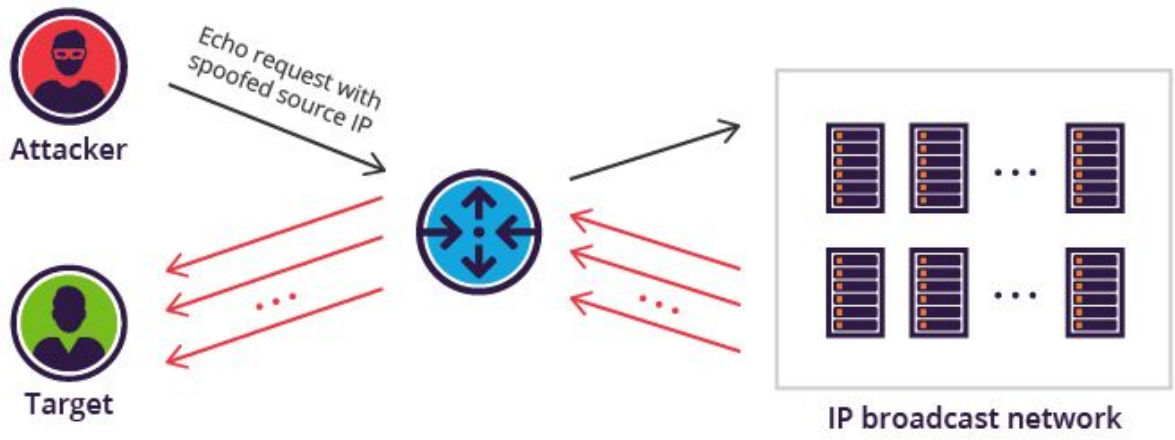
ICMP flooding

A ping flood is a simple denial-of-service attack where the attacker overwhelms the victim with ICMP "echo request" (ping) packets. This is most effective by using the flood option of ping which sends ICMP packets as fast as possible without waiting for replies. Most implementations of ping require the user to be privileged in order to specify the flood option. It is most successful if the **attacker has more bandwidth than the victim**. The attacker hopes that the victim will respond with ICMP "echo reply" packets, thus consuming both outgoing bandwidth as well as incoming bandwidth. If the target system is slow enough, it is possible to consume enough of its CPU cycles for a user to notice a significant slowdown.

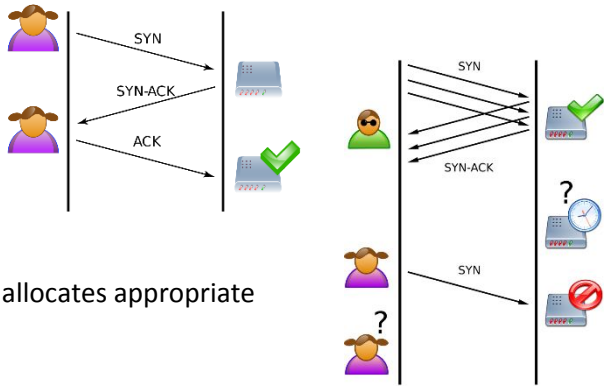
Smurf attack

A Smurf attack scenario can be broken down as follows:

- Smurf malware is used to generate a fake Echo request containing a spoofed source IP, which is actually the target server address.
- The request is sent to an intermediate IP broadcast network.
- The request is transmitted to all of the network hosts on the network.
- Each host sends an ICMP response to the spoofed source address.
- With enough ICMP responses forwarded, the target server is brought down.



SYNC Flood attack



Receiver allocates resources for each connection.
With enough requests, receiver can run out of resources.
Defense:
TCP Sync cookies -> modified TCP communication protocol.
Server sends a cookie inside the SYNC-ACK message. And opens the port again for another communication.
If he receives the same cookie inside the ACK message,, then he allocates appropriate recourses for the connection.

Volumetric and reflection attacks

A **volumetric attack** sends a high amount of traffic, or request packets, to a targeted network in an effort to overwhelm its bandwidth capabilities. These attacks work to flood the target in the hopes of slowing or stopping their services. Typically, request sizes are in the 100's of Gbps; however, recent attacks have scaled to over 1Tbps.

Similarly, to Linux where everything is an object, on Windows, everything is an object. are those that yield the largest bandwidth impacts. In a reflective attack, an attacker sends out small requests to a number of legitimate services hosted on the Internet. Their one key trick, they make it look like the request originated from the victim, and they use a request that commands a big response. As a result, they send out a stream small requests to services on the Internet, such as DNS and NTP, and then, in turn, those services send large responses to the victim. The transformation of small requests into large responses, is why it's called amplification.

Volumetric attacks predominantly rely on reflective techniques, but this is not always the case when an attack is sourced from a botnet. If a botnet is large enough, it does not need to go through another amplifying server, instead each compromised device in the botnet sends its traffic directly to the victim.

Common volumetric attack: ICMP Flood

Detection & mitigation of DOS

DDoS mitigation is a set of techniques or tools for resisting or mitigating the impact of distributed denial-of-service (DDoS) attacks on networks attached to the Internet by protecting the target and relay networks.

The first thing to do in DDoS mitigation is to identify normal conditions for network traffic by defining "traffic patterns", which is necessary for threat detection and alerting. DDoS mitigation also requires identifying incoming traffic to separate human traffic from human-like bots and hijacked web browsers. The process is done by comparing signatures and examining different attributes of the traffic, including IP addresses, cookie variations, HTTP headers, and JavaScript footprints.

After the detection is made, the next process is filtering. Filtering can be done through anti-DDoS technology like connection tracking, IP reputation lists, deep packet inspection, blacklisting/whitelisting, or rate limiting.