Elia Nicolaou EPL326 Final Exam

Introduction to Security

What is software security?

is an idea implemented to protect software against malicious attacks and other hacker risks so that the software continues to function correctly under such potential risks. Security is necessary to provide integrity, authentication, and availability.

Implicit Assumption:

* αν κάνουμε ένα recall στο αρχικό μας πρόβλημα της Αλίκης και του Bob. O Oscar μπορεί να προσπαθήσει να εισβάλει να δει τα μηνύματα που ανταλλάσσονται. Αυτό που προσθέσαμε ήταν ένα secure channel ώστε να μπορεί να αποτρέψει τον Oscar από το να εισβάλει => ο τρόπος που το έκανε ήταν με ένα αλγόριθμο κρυπτογράφησης.

τί είναι το θέμα όμως? υποθέσαμε ωστόσο ότι στα μέρη του encrypt / decrypt ο κώδικας μας τρέχει σωστά (it runs correctly – it uses memory correctly) ωστόσο κανείς δεν μας απέδειξε ότι όντως ο κώδικας τρέχει απόλυτα σωστά, ή το πώς όντως γίνετε η χρήση του memory, πώς το software τρέχει ή πως πρέπει να διαχειριστούμε τυχόν λάθη.

=> a good crypto technique ensures you that u know how to handle the memory correctly.

Software RoadMap:

 is an idea used by engineering teams to provide a high-level overview of the software development process.
=> software properties (show us what is happening in the memory)
=> software bugs (what happens if the program crash, how to handle it, what if someone tries to take advantage)
=> attacks and exploitation
=>Defenses

Software:

software us actually a collection of instructions and data that tell the computer how to work.
The programs are usually written in a high-level language (c,c++,java)
→ they are easier and more efficient for programmers because they are closer to natural languages than the machine languages.
→ high-level languages are translated into machine language using a compiler or an interpreter or a combination.
so overall, programs are compiled for execution: **προσπαθούνε να βρούνε μια μηχανή η οποία θα εκτελέσει το πρόγραμμα.**
 1. Machine Code (unmanaged & unsafe code): γιατί όμως το θέλουμε να το χρησιμοποιούμε : επειδή διότι θέλω να διαχειριστώ την μνήμα, (δεν κάνουν έλεγχο – χειρισμό μνήμης) => 1. performance 2. ελευθερία ώστε να μην χάνεις τα διάφορα functionalities.
 2. Virtual Machine Code such as JVM (managed and safe code).

* different architectures exhibit different properties in executing software (some generic concepts apply to all).

Safe vs unsafe systems

Safe programming systems:

- execute managed code in some virtual machine (e.g. Java => bytecode/vm) (εικονική μηχανή με ελεγχόμενη μηχανή)
- perform static analysis and compile time – this idea of rejecting the unsafe code and use run-time checks (δίνει την ιδέα κάποιας περιορισμένης πρόσβασης)

Unsafe programming systems:

- gives unrestricted memory access (doesn’t limit your functionalities, δίνει καλύτερο performance & δεν περιορίζονται τα διάφορα functionalities.
- Low-level accessing => σου παρέχει ελευθερία.
- legacy code

Software Composition

Elia Nicolaou EPL326 Final Exam

Data:
Εάν δεν διαχειρίζεσαι σωστά τα δεδομένα σου μπορείς να φτάσεις σε φοβερά προβλήματα.
-> Τα δεδομένα είναι static pre-defined and allocated at compile-time (τα ξέρει ο compiler).
-> Dynamically allocated and short-lived (usually at the Stack) -> τα ξέρει ο compiler.
-> Dynamically allocated and long-lived (usually at the heap) -> τα δεδομένα τα μαθαίνει στην πορεία πχ κάποια είσοδος από τον χρήστη.
Code:
known at the compile-time (ahead of time)
generated at run-time (just in Time) using a JIT engine.

Compilation, Loading and Execution steps:

compilation = the compiler prepares the program, it creates a binary file stored in the disk. This binary file contains code and data and has different sections. The binary might also have additional dependencies (some shared libraries). (compiler like c)
Loading = the loader reads the binary and maps it to a memory. This resolves different symbols and loads shared libraries.
Execution = the OS creates a structure called process (είναι εικονική δομή ενός εκτελέσιμου), a process is virtual concept of an executing program. A binary has a different layout (heap-stack) from an executing process mapped in a memory.

Processes

-> programs are executed with the help of the OS.
-> The OS creates an internal structure (a virtual process) for running the program.
-> A virtual process has several things => resources , state, a pointer to the next instruction.
* each process is executed as they are alone. (δεν ξέρουν τι γίνεται στην μνήμη).
  The OS execute several processes, concurrently. (virtual pages)

Virtual Memory

- a process works using memory, memory is given by the OS in a number of virtual pages (a few kilobytes each)
mapping = physical memory is mapped to virtual pages, every process has its own map (page tables)
if physical mem runs out, it is swapped to disk.
- not pages are loaded at once, code pages are loaded when needed (using some page faults)
  and some additional pages for data are allocated upon request. ***(φορτώνονται ουσιαστικά σταδιακά, πως καταλαβαίνω την φόρτωση = page table)***

a good practice is to have different memory areas => ο κώδικας να αποθηκεύεται σε διαφορετικά μέρη της μνήμης.
- code region has pages having code
- data region has pages having data
- stack has pages hosting temporary data
- heap has pages with dynamically allocated data

-> pages have permissions (some are enforced by hardware MMU / NX-bit κάποιες δεν μπορείς να τες εκτελέσεις καν)

Τι προβλήματα μπορεί να υπάρχουν στο κώδικα?
-> maybe it has an out of bound access => nobody will tell u that this is wrong, ωστόσο αυτό το πρόβλημα μπορεί να προκαλέσει μεγάλη ζημιά.

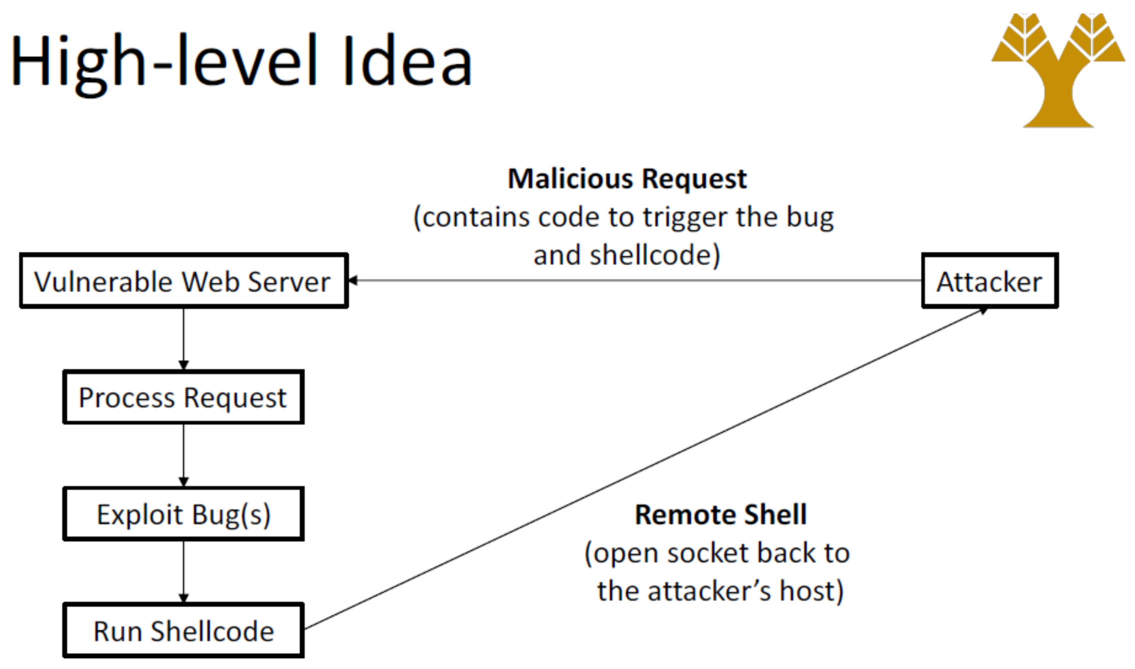
Vulnerability and Exploit

Vulnerability => is known as a software error (aka a bug) that can potentially allow someone to take advantage of the vulnerable program. (δίνει την ιδέα του να μπορώ να στέλνω διάφορα inputs και να εκμεταλλεύομαι το πρόγραμμα).
vulnerability = is a weakness which can be exploited by a threat actor, such as an attacker, to cross privilege boundaries within a computer system. To exploit a vulnerability, an attacker must have at least one applicable tool or technique that can connect to a system weakness.

Exploit => it is the process of controlling a program by taking advantage of one or more vulnerabilities. Not all vulnerabilities though, van be exploited. Exploitation is actually the next step in an attacker's playbook after finding a vulnerability.

Arbitrary Code Execution: the state of an exploit where an attacker can execute a program of their choice.
Shellcode: a machine code that a vulnerable program executed and serves the purposes of the attacker.
spawn a shell, download malware… (can do a lot of things)
- it's a heavily architecture dependent.

Software Exploitation: (logic)
The program of a user has some vulnerabilities (some bugs that someone can take advantage of). This program can be a program that is executing in the user-space or a program that is in the OS.
Cause of the vulnerabilities the user can become a victim of an attacker. The bugs of the program can be triggered using some malicious inputs, inputs can be sent over the network or can be sent locally.
*The point is that triggering bugs can lead to arbitrary code execution, which can run the shellcode.



Control Flow Attacks #letsdosomemoredangerousStaff

Elia Nicolaou EPL326 Final Exam

Functions
we know that software is composed by several functions (parts of the code that we can easily call)
function give the opportunity for code to be re-use. In a program a function can call a function and then another function = this is function chaining is called control-flow

whenever a function is called the flow **of the program is changed** (however the function must know where to return – how we save the address of the next instruction in the stack).
the change of the control flow must be done transparently, once the function is finished the control flow must find the way to resumed.
(γενικά πάντα έχουμε την διαίσθηση ότι τα πράγματα / οι εντολές εκτελούνται σειριακά αλλά όταν έχουμε συναρτήσεις έχουμε συνεχώς jumps)

- functions may have arguments, may return data, or may create local data => that means they want memory.

***we have the callee, and the caller (call site)**

Stack (δομή δεδομένων)

- functions need memory for their work. (stack)
- This memory is for short lived data, that means once the function is finished it job, we can easily get rid of any data involved.
- Stack is architecture dependent. The main idea does not change, ακολουθείται πάντα η ίδια λογική του push,pop και χρειάζεται να γνωρίζεις πάντοτε την κεφαλή.
- Stack is capable for holding several things, function arguments, the return address, the old frame pointer and local arguments. (υπάρχει κάποιος συνειρμός με τα πιάτα)
- Γενικά για να περιγράψω τον stack χρειάζεται να ξέρω απλά και μόνο την κεφαλή.

Stack of Intel (32-bit) the one that we are interested about.

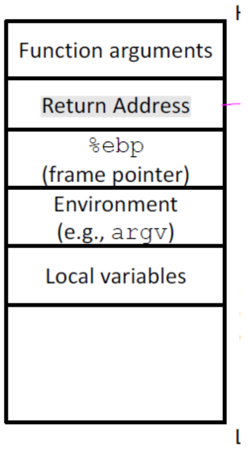
- The stack has the property that grows from higher-memory addresses to lower-memory addresses
- The top of the stack is always kept in a hardware register **(%esp)** αυτό πάει να πει ότι η κεφαλή πάντα φυλάγεται σε αυτόν τον καταχωρητή, αν αλλάξεις από μόνος σου τότε αλλάζεις την διεύθυνση της μνήμης.
- Each function creates a new stack frame upon executing, and the stack frame is destroyed once the functions is finished.
- The top of the stack frame is kept in a hardware register **(%ebp)**

ebp = stack frame pointer
esp = top of the stack pointer

Stack insertion = push | βάζεις μια καινούρια κεφαλή, κατά 4 bytes, επειδή πάμε από ψηλότερες σε χαμηλότερες διευθύνσεις αυτό πάει να πει ότι αφαιρώ 4.

Stack deletion = pop | βγάζεις την κεφαλή, κατά 4 bytes, επειδή πάμε από ψηλότερες σε χαμηλότερες διευθύνσεις αυτό πάει να πει ότι προσθέτω 4. (0xFF) means is deleted.

the pictures show the stack frame in Intel 32-bit =>



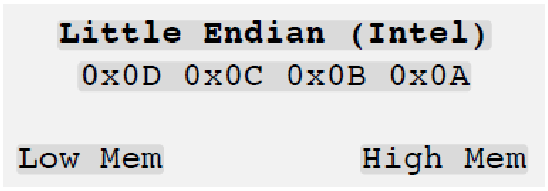
Elia Nicolaou EPL326 Final Exam

* προτιμάμε 1000 φορές να κρασάρει το πρόγραμμα παρά κάποιος να του επιτεθεί, αν γράψεις την μνήμη εντελώς randomly τότε το πρόγραμμα ναι θα κρασάρει.

Endianness

Γενικά υπάρχουν δύο τρόποι να γράψεις / αποθηκεύσεις στην μνήμη.
α) Little Endian
b) Big Endian

ο Intel χρησιμοποιεί την λογική του Little Endian, (Low Mem->High Mem)



Πως γίνεται ουσιαστικά ένα control-flow attack?

Εκμεταλλεύομαι την δυνατότητα να γράψω εγώ κάτι μέσα στην μνήμη, θα γράψω όπως ακριβώς θέλω εγώ προσθέτω ουσιαστικά ένα malicious input. Προσπαθώ να βάλω inputs που κάνουν bypass διάφορα checks. Παίζει πολύ το γεγονός του που επιστρέφω.

Control flow Attack overall:

we know that the memory of some process contains some control data. (return addresses). You can get as example, that the return address is stored in the stack.
Control data dictate the flow of the program , overwriting is possible since control data are co-located with other buffers that can be overwritten due to program's vulnerabilities.

-> why the control flow attack is not perfect:

it is not the best attack that we could do, stack was for surely not handled correctly and the program crashes in the end. Δεν φροντίσαμε μετά να πάει καλά.
It is easy to carry out: just change the value of the return address,
goal achieved. (although is dirty)
-> ήταν πολύ εύκολο να γίνει.

***all the idea is based on finding an input that can bypass the checks.**

possible bug : strcpy (μπορώ να γράψω εκτός του array, θα γράψω ότι θέλω) => όλη η ιδέα βασίζεται στο πως θα κάνει compute το user input.

Elia Nicolaou EPL326 Final Exam

Shellcode

What is shellcode: πλέον θα προσπαθήσουμε να δώσουμε ένα input, που ουσιαστικά θα προσπαθεί να μας ανοίξει ένα shell, θέλουμε να κάνουμε/πείσουμε το πρόγραμμα να εκτελέσει το δικό μας κώδικα. Άρα ουσιαστικά αρχικά πρέπει να προσπαθήσουμε να φτιάξουμε αυτόν τον κώδικα.
Το shellcode πρέπει να βρίσκεται σε μορφή machine code -> το input που θα δώσουμε θα είναι machine code.

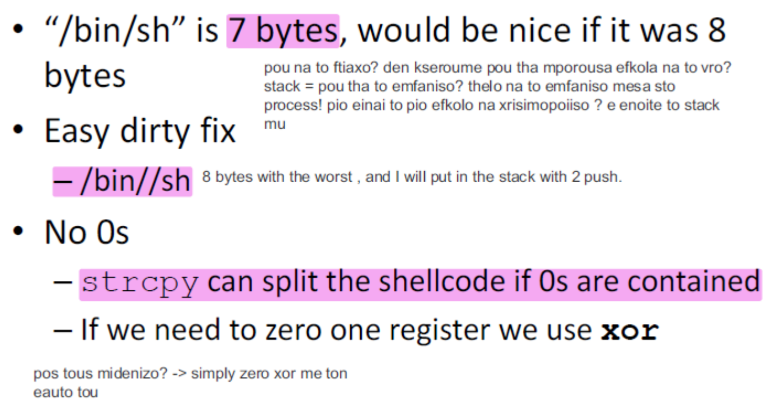
* Is actually a program, a very small program, that is stuffed in a user input (spawns a remote shell, downloads malware, creates a user and etc).
Shellcode is highly architecture dependent (32bit) and its highly unorthodox programming involved, δεν είναι δομημένος προγραμματισμός. (custom assembly).

Why small? the program is usually small and to the point, especially in buffer overflows, the vulnerable buffers may have limited size**. No \0s allowed,** otherwise string copies can destroy the shellcode (there is no flexibility)

Spawning a shell: για να ανοίξω ένα shell συνήθως πρέπει να καλέσω ένα system call, execve, αντιστοιχεί το παλιό τρέχω process με ένα νέο.

More about system calls: τα καλούμε με μια συγκεκριμένη εντολή, την int, διακόπτει το τι γίνετε και εκτέλεσε το shell.
-> can be invoked using a software interrupt, the assembly instruction is int **(for execve the interrupt num is 11)**
-> The OS has a system call table, each system call invokes the appropriate code.

hacks -> no 0s, strcpy can split the shellcode If 0s are contained, if we need to zero one register we use xor.


- "/bin/sh" is 7 bytes, would be nice if it was 8 bytes
- Easy dirty fix
  - /bin//sh
- No 0s
  - `strcpy` can split the shellcode if 0s are contained
  - If we need to zero one register we use **`xor`**

pou na to ftiaxo? den kseroume pou tha mporousa efkola na to vro? stack = pou tha to emfaniso? thelo na to emfaniso mesa sto process! pio einai to pio efkolo na xrisimopoiiso ? e enoite to stack mu

8 bytes with the worst , and I will put in the stack with 2 push.

pos tous midenizo? -> simply zero xor me ton eauto tou

Elia Nicolaou EPL326 Final Exam

Code Injection

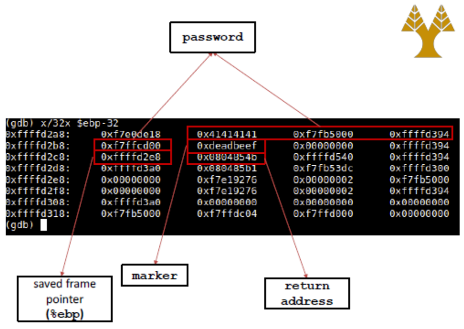
ret = κάνει jump, ο έλεγχος περνά σε εμάς όταν τελειώσει η authenticate root για να κάνω debug ουσιαστικά θα πάω να βάλω κάποιο έλεγχο πάνω / one breakpoint πάνω στο ret.

Stack

->όπως έχουμε πει στο stack μας περιέχεται το control data, αυτό πάει να πει ότι εκεί αποθηκεύεται και το return address. Επομένως το επόμενο που μπορεί να καταλάβει κανείς είναι ότι μπορεί να γίνει hijack.
-> also function pointer and Vtable pointers allocated locally.
-> we have overflows in local buffers, that can overwrite control data (known as control- flow attacks)

-> seeing the stack u can understand lots of things, like the saved frame pointer, the marker, or the return address.

***we should try understanding where an attack can be placed.**



```
(gdb) x/32x $ebp-32
0xffffd2a8:     0xf7e0de18      0x41414141      0xf7fb5000      0xffffd394
0xffffd2b8:     0xf7ffcd00      0xdeadbeef      0x00000000      0xffffd394
0xffffd2c8:     0xffffd2e8      0x0804854b      0xffffd540      0xffffd394
0xffffd2d8:     0xffffd3a0      0x080485b1      0xf7fb53dc      0xffffd300
0xffffd2e8:     0x00000000      0xf7e19276      0x00000002      0xf7fb5000
0xffffd2f8:     0x00000000      0xf7e19276      0x00000002      0xffffd394
0xffffd308:     0xffffd3a0      0x00000000      0x00000000      0x00000000
0xffffd318:     0xf7fb5000      0xf7ffdc04      0xf7ffd000      0x00000000
(gdb)
```

5

Buffer overflow

* take advantage of a bug, that can write over the stack.
use of write primitive can overwrite control-data as the return address to.

πως μπορώ να κάνω defend το return address?

-> με κάποιο τρόπο πρέπει να προσπαθήσουμε να το κάνουμε defend, εδώ έρχεται η λειτουργία του canary, είναι μια τυχαία τιμή, εκεί που είχαμε το marker.
Αν τώρα κάποιος προσπαθήσει να γράψει αυτό πάει να πει ότι θα γράψει και πάνω από το canary.

Τα canary values are generated randomly upon process creation and they are stored in a global variable (%gs is involved usually).

<u>*nice, so how the canary value is helpful for defending our program? </u>-> all functions epilogues are modified, we going to check if the canary values had been modified before returning back.

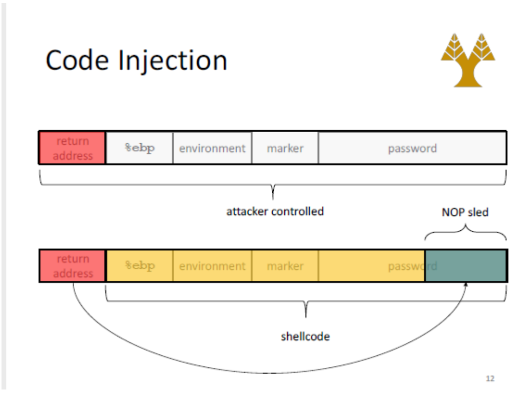
can we bypass canaries? yes, we can, but is a very cheap solution but is better to use information leaks and read primitives. *θεωρείται κάπως ξεπερασμένο πλέον.

So, what is code injection?

Code injection is the exploitation of a computer bug that is caused by processing invalid data. The injection is used by an attacker to introduce code into a vulnerable computer program and change the course of execution.
We inject code to data, code is contained in a malicious input.
-> we can overwrite control data, so the new control data jump inside the malicious input and the new code, the malicious one is executed.

Defense to the code injection:

The NX bit (no-execute) is a technology used in CPUs to segregate areas of memory for use by either storage of processor instructions (code) or for storage of data. The processor will then refuse to execute any code residing in these areas of memory.
although -> ROP, change of permissions can attack the nx-bit.



Elia Nicolaou EPL326 Final Exam

Return-oriented Programming (ROP)
-> έχουμε φτάσει στο σημείο ότι πλέον τα δεδομένα μας δεν είναι εκτελέσιμα, άρα αυτό είναι ένα μεγάλο πρόβλημα για τον attacker.
τι πρέπει να κάνω? ιδανικά θα θέλω να αλλάξω τα permissions.

*αφού τα δεδομένα μας πλέον δεν είναι εκτελέσιμα, stack is not executable, we have to find another way..., αν επιχειρήσουμε να κάνουμε exploit με τέτοιο τρόπο θα κρασάρει το πρόγραμμα μας.

Non – executable memory:
several names that say that a memory is not executable is:
- NX-bit (non-executable bit)
- DEP (data execution prevention)
- W^X (Write xor execute).

-> all this are enforced in hardware (MMU)
-> memory pages cannot be executable and writable at the same time, stack & heap are writable but not executable, code is executable but not writable.

how to change permissions ? using some system calls, mprotect () for linux / virtualprotect windows

Challenge: to find where to return, if I return to the stack this is going to be a problem as for one more time stack is not executable. So, smart idea we have to return to the only executable part which is the .text. (υπάρχει πολύς κώδικας που μπορώ να εκμεταλλευτώ).

ROP: θα χρησιμοποιήσουμε τα λεγόμενα gadgets, θα γεμίσω το κώδικα μου με τα gadgets.
-> all the idea of rop is based on the code reuse : επαναχρησιμοποιούμε τον κώδικα που έχουμε.
Programs include large parts of existing code, which they are available during exploitation.

Gadgets: are small sequences of instructions ending with a ret. They execute some code and return.
we take advantage of stack that is attacker – controlled, we are going to use %esp as the program counter.
chaining several gadgets: (rop) , Turing complete, and in practice used to make a region executable.
Κάθε gadget τελειώνει με το ret.

***πλέον γεμίζουμε το buffer με διευθύνσεις κώδικα προγράμματος.**

Intel & CISC architecture
->Dense instruction set => all values map to a valid instruction.
-> Non-aligned instructions.
-> variable-length instructions (jumping in the middle of an instruction generates some new instructions).

Defending ROP:
well , rop is based in exact knowledge of the code layout. Code address are the begging of the ROP gadgets.
-> good idea to randomize stack. Πώς θα κλείσω/κρύψω τον κώδικα μου?

=> address space layout randomization (ASLR).
=> position independent code. (works with offsets)

Elia Nicolaou EPL326 Final Exam

Information Leaks:

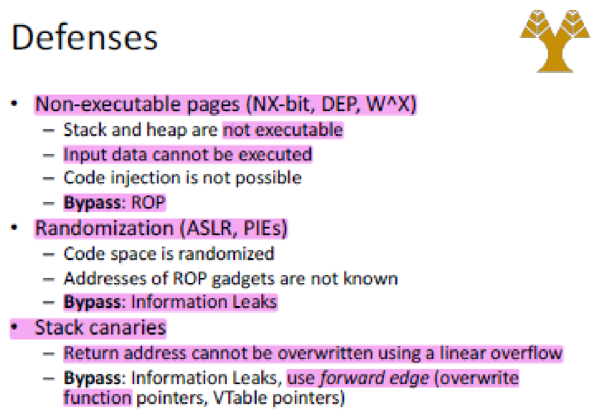
information leaks are some bugs that let you read the process layout. So that means that ASLR is not such a good defend.

ASLR -> revealing the address where a shared library is mapped is enough, all gadgets are just moved to a new offset.

Stack canaries -> revealing the contents of the stack is enough, the canary is stored in the stack.

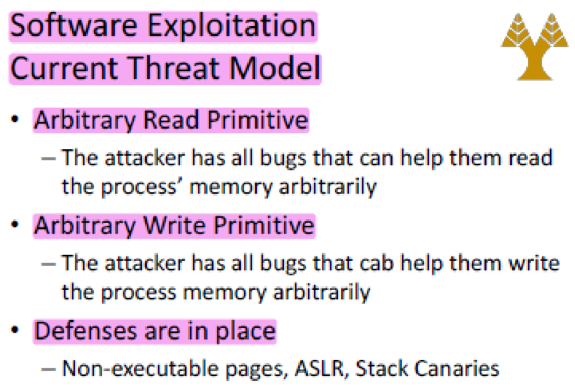
information leaks can be used repeatedly in order to uncover the layout of the process.

Web browsers- -> bugs can be abused by JavaScript, a malicious JavaScript can use an information-leak bug programmatically.



Defenses

- Non-executable pages (NX-bit, DEP, W^X)
  - Stack and heap are not executable
  - Input data cannot be executed
  - Code injection is not possible
  - **Bypass**: ROP
- Randomization (ASLR, PIEs)
  - Code space is randomized
  - Addresses of ROP gadgets are not known
  - **Bypass**: Information Leaks
- Stack canaries
  - Return address cannot be overwritten using a linear overflow
  - **Bypass**: Information Leaks, use *forward edge* (overwrite function pointers, VTable pointers)



Software Exploitation Current Threat Model

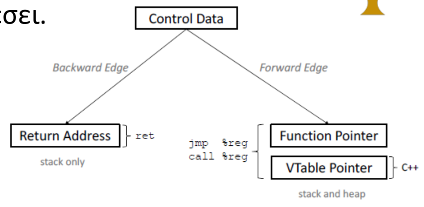
- Arbitrary Read Primitive
  - The attacker has all bugs that can help them read the process' memory arbitrarily
- Arbitrary Write Primitive
  - The attacker has all bugs that cab help them write the process memory arbitrarily
- Defenses are in place
  - Non-executable pages, ASLR, Stack Canaries

Elia Nicolaou EPL326 Final Exam

Heap Exploitation

είχαμε αναφέρει ότι το control data, είναι δεδομένα που βρίσκονται στην μνήμη του προγράμματος και αποφασίζουν πως θα εξελιχθεί η ροή.
το control data χωρίζεται σε δύο μέρη

backward edge = περιέχει τα δεδομένα αυτά που δείχνουν που να επιστρέψει το πρόγραμμα.
forward edge = δείχνου που πρέπει να κάνει jump το πρόγραμμα, που να πάει να εκτελέσει.



Heap:

heap is much more complicated than the stack, there is not some interface.
-> heap is a memory area that sores resources dynamically.
-> is managed by a user-space library (standard libc)
-> API access (allocator) -> βάζω κάτι μέσα στο heap – δεσμεύω μνήμη.
(malloc,free,realloc)
-> no cpu support unlike stack.

Heap overflows:

-> overwrite control in the heap , unlike stack we don’t have any return addresses in the heap. We have only Function pointers and Vtable pointers.
-> the above point makes heap much harder for exploit writing

- rop uses the stack for executing the gadgets, στο heap δεν έχουμε gadgets.
- stack pointer used as a program counter and there is no stack in the heap. Executing a single gadget will transfer control to the stack when the first **ret** executes.

τότε τι θα κάνω ???? => το τι θέλουμε να γίνει είναι με κάποιο τρόπο να βάλουμε το stack in heap.

Stack Pivoting:

A common method for an attacker to control program execution involves creating a "fake stack" using attacker-specified values. When the attacker tricks the victim computer into using a fake stack, the attacker can control the program execution, because the call stack specifies the return behavior of the program from its current state.

Return addresses are important because when the computer executes a "ret" instruction, it basically loads the value at the address pointed to by the ESP register into the EIP register, and makes ESP point to one position lower on the stack. Semantically this means that ESP points to the top of the stack, and the computer "popped" the value off the top of the stack into EIP, which is the instruction pointer register. Another way to say this is that execution "returned" to the address stored at the top of the call stack. This fact is crucial to the way ROP exploits work.

* πρόκειται να αλλάξουμε την διεύθυνση του stack pointer. (xchg %eax, %esp)

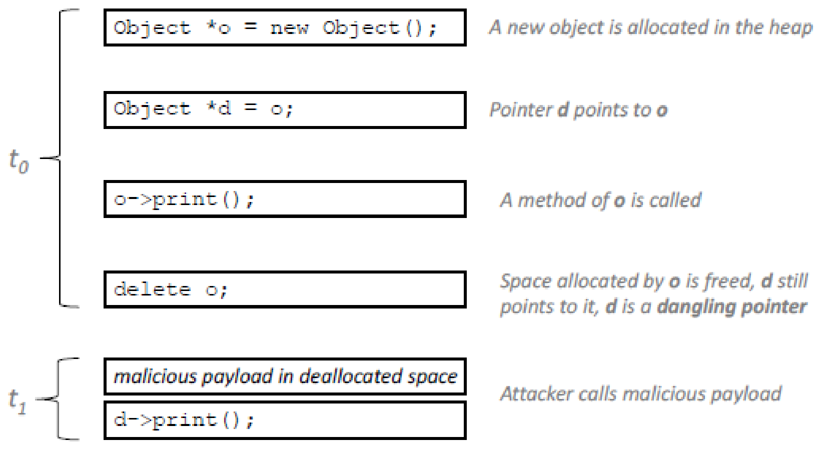
Spatial vs Temporal Safety:

Spatial Safety: The goal of spatial safety is to ensure that every memory access occurs within the bounds of a known object. Spatial safety is typically enforced by inserting runtime checks before pointer dereferences.
->overflow attacks παραβιάζουν το safety αυτό.
->it is based on data geometry and it is hand y for stacks.

Temporal Safety: Temporal memory safety is an orthogonal problem to spatial memory safety Abuse data access over time, place malicious data in de-allocated spaced and trigger dangling pointers that still point to it.

Elia Nicolaou EPL326 Final Exam

Use – after – free:



```
Object *o = new Object();
```
A new object is allocated in the heap

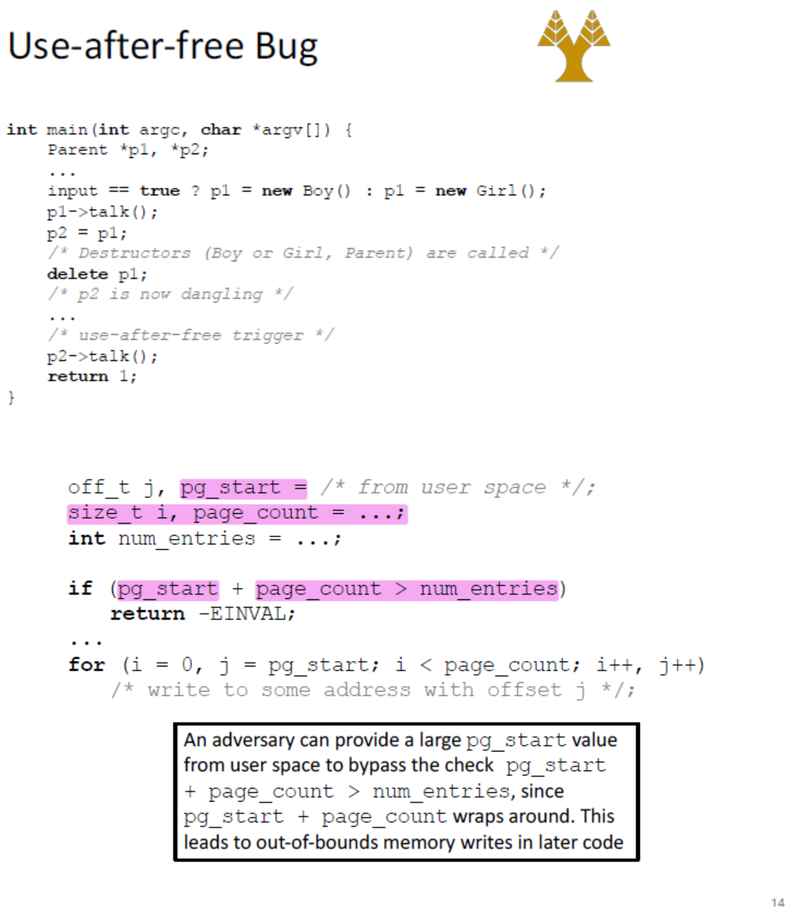
```
Object *d = o;
```
*Pointer **d** points to **o***

```
o->print();
```
*A method of **o** is called*

```
delete o;
```
*Space allocated by **o** is freed, **d** still points to it, **d** is a **dangling pointer***

malicious payload in deallocated space
```
d->print();
```
Attacker calls malicious payload

Θα προσπαθήσω να αναγκάσω το πρόγραμμα να κάνει συνέχεια allocations. (slides!!)



Use-after-free Bug

```
int main(int argc, char *argv[]) {
    Parent *p1, *p2;
    ...
    input == true ? p1 = new Boy() : p1 = new Girl();
    p1->talk();
    p2 = p1;
    /* Destructors (Boy or Girl, Parent) are called */
    delete p1;
    /* p2 is now dangling */
    ...
    /* use-after-free trigger */
    p2->talk();
    return 1;
}
```

```
off_t j, pg_start = /* from user space */;
size_t i, page_count = ...;
int num_entries = ...;

if (pg_start + page_count > num_entries)
    return -EINVAL;
...
for (i = 0, j = pg_start; i < page_count; i++, j++)
    /* write to some address with offset j */;
```

An adversary can provide a large `pg_start` value from user space to bypass the check `pg_start + page_count > num_entries`, since `pg_start + page_count` wraps around. This leads to out-of-bounds memory writes in later code

14

Elia Nicolaou EPL326 Final Exam

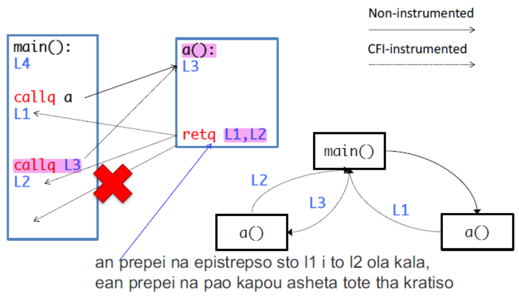
Program Analysis and Applications

=> θέλουμε να δούμε με ποιο τρόπο μπορούμε να δημιουργήσουμε άμυνες οι οποίες θα κάνουν την ζωή του επιτιθέμενου πιο δύσκολη.

Modern Software Hardening. (να μπορείς να αμύνεσαι να μπορείς να βρίσκεις τα διάφορα bugs).
there are some techniques for defending the hardware against some attacker that has arbitrary read/write capabilities.

=> the good thing is that those techniques can be applied directly to the binaries when the source is not available.
Legacy software.
=> can also be applied to source code, needs software re-compilation.
=> there are although some performance overhead, εάν το πρόγραμμα μας είναι αργό αυτό δεν είναι κάποια ελκυστική τεχνική.

Control-flow Integrity (CFI) => η λογική του είναι να αναλύσει το στατικό flow που υπάρχει.
-> πιο πλούσια διαδικασία μπορεί να κάνει περισσότερα πράγματα.


Non-instrumented
CFI-instrumented
main():
L4
callq a
L1
callq L3
L2
a():
L3
retq L1,L2
main()
L2
L3
L1
a()
a()
an prepei na epistrepso sto l1 i to l2 ola kala,
ean prepei na pao kapou asheta tote tha kratiso

Control-flow integrity (CFI) is a general term for computer security techniques that prevent a wide variety of malware attacks from redirecting the flow of execution (the control flow) of a program.

Problems of CFI:

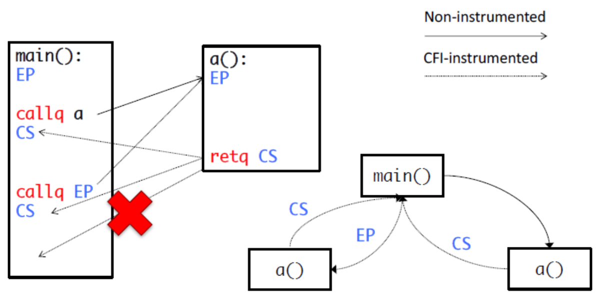
- is hard to compute perfect control-flow-graph

-> there is no source code
->loadable modules
->dynamic code

- performance is a major problem.

Coarse-grained CFI (2 labels)

* το πιο καλό απλό που μπορείς να κάνεις είναι να μειώσεις τα label, αυτό κάνει την κατάσταση πάρα πολύ δύσκολη για τον attacker. (one entry-point , and a call site).



Deployed CFI

-> it is supported by modern compilers. (CFG Microsoft)
-> it has hardware support.
Intel has announced hardware instruction for CFI
Shadow stacks (return addresses are stored in a separate h/w memory.

Program Analysis:

static: (δεν βλέπεις πως εκτελείται όταν μια είσοδος μπαίνει στον κώδικα)
-> it is before execution
-> source based / binary based
-> there is no adaptation to particular inputs.
dynamic: (take as an example the gdb analysis) – βλέπω τι γίνεται ανά πάσα στιγμή.
->happens during execution
->can be adapt to certain inputs.

Elia Nicolaou EPL326 Final Exam

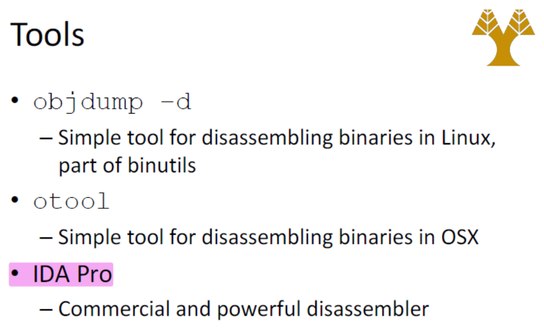
Static Analysis:

the idea of static analysis is to analyze the source of the program but without executing it (no inputs)
-> the source can be in a high-level language, or in bytecode , or machine code.

the source based static analysis usually is performed at the compiler-level.
LLVM (low lever virtual machine) -> written in bytecode.
-> compiler infrastructure that allows to add custom passes
-> all phases in the compilation are represented using LLVM IR(intermediate representation).

Binary Analysis: (we are trying to see and change the binary)

-> we are disassembling a binary for analysis (open problem x86)
we have variable length instructions and data are mixed with the code.
->recursive disassembly, follow jumps and disassemble targets.
-> linear disassembly.



Tools

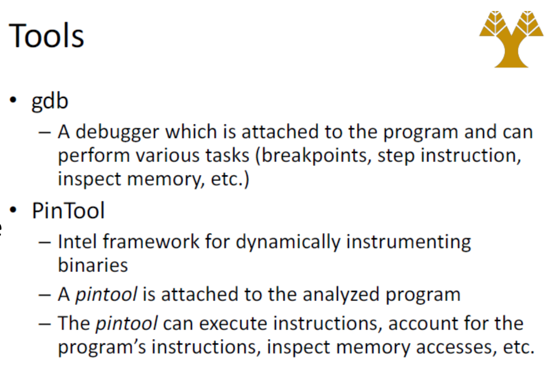
- `objdump -d`
  - Simple tool for disassembling binaries in Linux, part of binutils
- `otool`
  - Simple tool for disassembling binaries in OSX
- IDA Pro
  - Commercial and powerful disassembler

Dynamic Analysis:

- it analyzes program while executing,
it is usually very slow, and analysis observes actual inputs.
* the analysis code runs in parallel with the program's code.

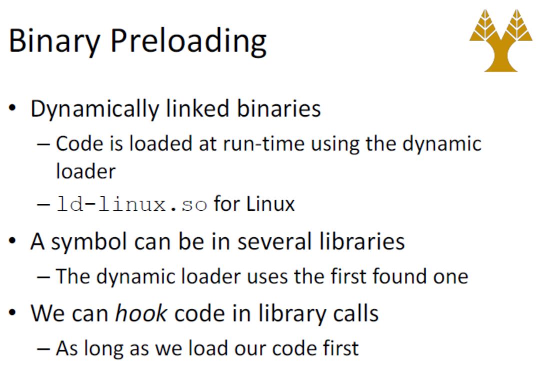
applications :

- program instrumentation
- bug finding (assisted fuzzing – finds lots of bugs)
- Malware identification (see if a downloaded program is malware or not.



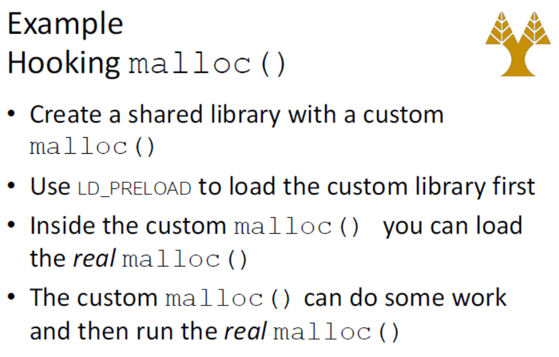
Tools

- gdb
  - A debugger which is attached to the program and can perform various tasks (breakpoints, step instruction, inspect memory, etc.)
- PinTool
  - Intel framework for dynamically instrumenting binaries
  - A *pintool* is attached to the analyzed program
  - The *pintool* can execute instructions, account for the program's instructions, inspect memory accesses, etc.



Binary Preloading

- Dynamically linked binaries
  - Code is loaded at run-time using the dynamic loader
  - `ld-linux.so` for Linux
- A symbol can be in several libraries
  - The dynamic loader uses the first found one
- We can *hook* code in library calls
  - As long as we load our code first



Example Hooking `malloc()`

- Create a shared library with a custom `malloc()`
- Use LD_PRELOAD to load the custom library first
- Inside the custom `malloc()` you can load the *real* `malloc()`
- The custom `malloc()` can do some work and then run the *real* `malloc()`

Elia Nicolaou EPL326 Final Exam

Introduction to network security (network founded at 1969)

the network it was a game changer:

we have software exploitation over the network that means local and remote (εξ αποστάσεως) attacks/ers.
Μπορείς να βρεις περισσότερους στόχους.
=> πρέπει να ξεχωρίσουμε ποιοι users, είναι καλοί στόχοι.
=> πώς γίνονται τα attacks στο network? έχουμε δύο είδη επιτιθεμένων τους active and passive attackers.

- protocols
- communication & applications \

=> το δίκτυο εισάγει περισσότερη πολυπλοκότητα αφού πολλοί παράμετροι επεξεργάζονται ταυτόχρονα, παίζουν ρόλο.

-> πλέον δεν υπάρχουν σύνορα, είναι παντού, υπάρχει μια πληθώρα από εφαρμογές αλλά έχουμε πλέον την έννοια του IoT όπου πολλές συσκευές αλληλοεπιδράνε ταυτόχρονα.

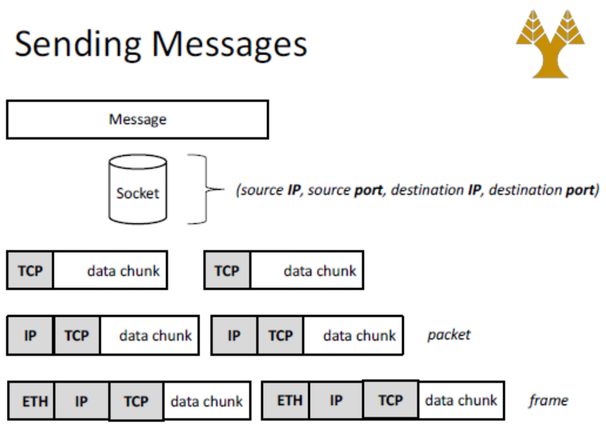
στην προκυμμένη περίπτωση του network security μας ενδιαφέρει κυρίως το μέρος του application layer που απαρτίζεται ουσιαστικά από το application,presentation & session.

Network communication: (πώς γίνεται η επικοινωνία)
η επικοινωνία γίνεται μέσω τον sockets σε άποψη λειτουργικού συστήματος, έχουμε κάποιους υπολογιστές και μέσα των sockets φτάνουμε σε κάποια arrangements.

Network communication, or internetworking, defines a set of protocols (that is, rules and standards) that allow application programs to talk with each other without regard to the hardware and operating systems where they are run. Internetworking allows application programs to communicate independently of their physical network connections.

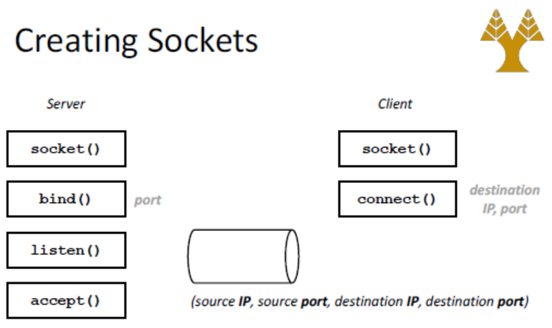
Sending Messages:

Start the connection.
Transfer data
End the connection.



Creating Sockets:

Creating a socket is in some ways similar to opening a file.
 This function creates a file descriptor and returns it from the function call. You later use this file descriptor for reading, writing and using with other socket functions
Step 1: Creating a socket:
Step 2: Binding an address and port number.
Step 3: Listen for incoming connections.
Step 4: Accepting a connection.

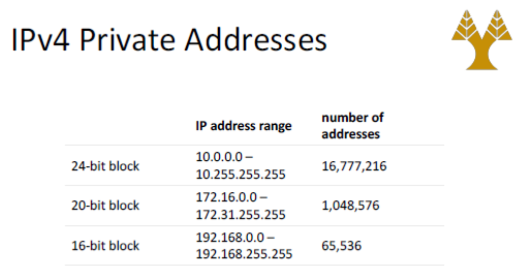


Elia Nicolaou EPL326 Final Exam

IP addresses:

The importance of IP addresses follows from the fact that each host on the Internet has a unique IP address. Thus, although the Internet is made up of many networks of networks with many different types of architectures and transport mediums, it is the IP address which provides a cohesive structure so that at least theoretically, (there are routing issues involved as well), any two hosts on the Internet can communicate with each other.

*not all IP addresses are routable -> there are some private addresses.



IPv4 Private Addresses

	IP address range	number of addresses
24-bit block	10.0.0.0 – 10.255.255.255	16,777,216
20-bit block	172.16.0.0 – 172.31.255.255	1,048,576
16-bit block	192.168.0.0 – 192.168.255.255	65,536

Address Resolution Protocol (ARP)

Address Resolution Protocol (ARP) is a protocol or procedure that connects an ever-changing Internet Protocol (IP) address to a fixed physical machine address, also known as a media access control (MAC) address, in a local-area network (LAN).

This mapping procedure is important because the lengths of the IP and MAC addresses differ, and a translation is needed so that the systems can recognize one another. The most used IP today is IP version 4 (IPv4). An IP address is 32 bits long. However, MAC addresses are 48 bits long. ARP translates the 32-bit address to 48 and vice versa.

=> what is actually do, it associates ethernet devices with IP addresses (A mac address is paired with an IP address)
=> each ethernet frame has a 48-bit address
=> ARP broadcasts an IP address (host with the Ip address responds with an IP/Ethernet address pair)

where is the problem? there is no authentication.

ARP spoofing: (πως μπορείς να το «κοροϊδέψεις»)

as we said before, ARP has no authentication, a malicious host may claim to have several IP addresses

An ARP spoofing, also known as ARP poisoning, is a Man in the Middle (MitM) attack that allows attackers to intercept communication between network devices. The attack works as follows:

1. The attacker must have access to the network. They scan the network to determine the IP addresses of at least two devices—let's say these are a workstation and a router.
2. The attacker uses a spoofing tool, such as Arpspoof or Driftnet, to send out forged ARP responses.
3. The forged responses advertise that the correct MAC address for both IP addresses, belonging to the router and workstation, is the attacker's MAC address. This fools both router and workstation to connect to the attacker's machine, instead of to each other.
4. The two devices update their ARP cache entries and from that point onwards, communicate with the attacker instead of directly with each other.
5. The attacker is now secretly in the middle of all communications.

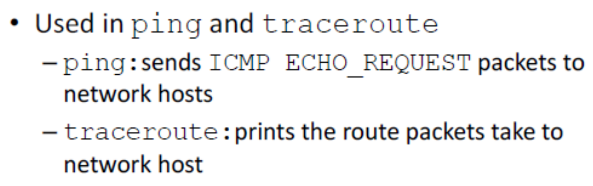
Defense of ARP spoofing:

1. Use a Virtual Private Network (VPN)—a VPN allows devices to connect to the Internet through an encrypted tunnel. This makes all communication encrypted, and worthless for an ARP spoofing attacker.
2. **Use static ARP—the ARP protocol** lets you define a static ARP entry for an IP address, and prevent devices from listening on ARP responses for that address. For example, if a workstation always connects to the same router, you can define a static ARP entry for that router, preventing an attack.
3. Use packet filtering—packet filtering solutions can identify poisoned ARP packets by seeing that they contain conflicting source information and stop them before they reach devices on your network.

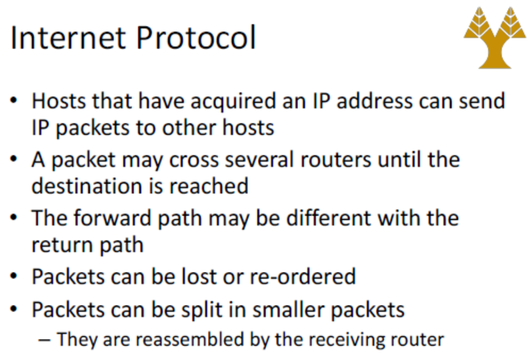
4. Run a spoofing attack—check if your existing defenses are working by mounting a spoofing attack, in coordination with IT and security teams. If the attack succeeds, identify weak points in your defensive measures and remediate them.

Internet Protocol & Internet Control Message Protocol (ICMP)

it is a protocol for sending error messages and operational information.



- Used in `ping` and `traceroute`
  - `ping`: sends `ICMP ECHO_REQUEST` packets to network hosts
  - `traceroute`: prints the route packets take to network host



Internet Protocol

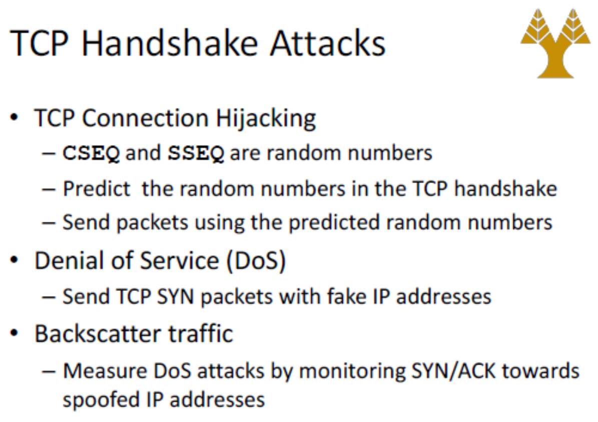
- Hosts that have acquired an IP address can send IP packets to other hosts
- A packet may cross several routers until the destination is reached
- The forward path may be different with the return path
- Packets can be lost or re-ordered
- Packets can be split in smaller packets
  - They are reassembled by the receiving router

Reliable communication:

TCP allows reliable communication between two end points.

TCP handshake Attacks:

TCP SYN flood (a.k.a. SYN flood) is a type of Distributed Denial of Service (DDoS) attack that exploits part of the normal TCP three-way handshake to consume resources on the targeted server and render it unresponsive.

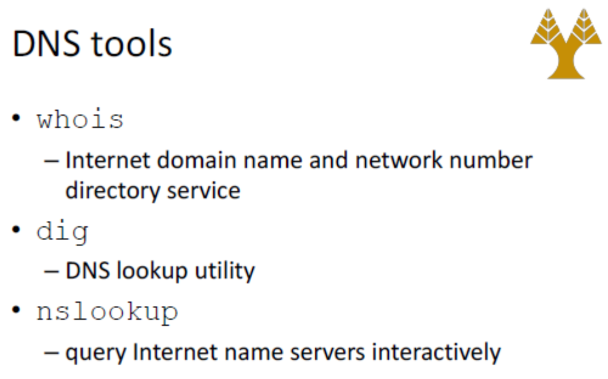


TCP Handshake Attacks

- TCP Connection Hijacking
  - **CSEQ** and **SSEQ** are random numbers
  - Predict the random numbers in the TCP handshake
  - Send packets using the predicted random numbers
- Denial of Service (DoS)
  - Send TCP SYN packets with fake IP addresses
- Backscatter traffic
  - Measure DoS attacks by monitoring SYN/ACK towards spoofed IP addresses

DNS:

is distributed tree-hierarchy, with mapping names to IP addresses.
there are several **DNS attacks**: the main goal of the attacks is to hijack a domain name and capture traffic.
phishing



DNS tools

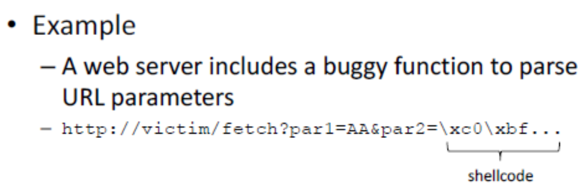
- `whois`
  - Internet domain name and network number directory service
- `dig`
  - DNS lookup utility
- `nslookup`
  - query Internet name servers interactively

Elia Nicolaou EPL326 Final Exam

Attacking and Defending the Network
Local and Remote Attacker :
Remote: A remote exploit works over a network and exploits the security vulnerability without any prior access to the vulnerable system. wget http://victim....
Local: A local exploit requires prior access to the vulnerable system and usually increases the privileges of the person running the exploit past those granted by the system administrator. Exploits against client applications also exist, usually consisting of modified servers that send an exploit if accessed with a client application. program 'printf "....

Remote Inputs:
programs can take inputs from the network, and the inputs are received using sockets.
some examples are a web server processes HTTP requests, or documents, a DNS servers processes DNS requests or an email sever processes.

Remote Exploitation:
Remote exploitation techniques are used to exploit a product or a component of a product by an attacker who does not have access to the computer being targeted.
-> the shellcode must be embedded in a network payload. The shellcode is not in argv[] but sent over the network.



- Example
  - A web server includes a buggy function to parse URL parameters
  - `http://victim/fetch?par1=AA&par2=\xc0\xbf...`

shellcode

Remote Attacker's Goal:
- Usually attacker's goal is servers as they usually contain some very valuable data.
- Hosts can be also a goal, an attacker can control several ordinary hosts. (bots). These bots comprise a Botnet.
- Lastly, users, they want to compromise massively users.

Botnets:
Botnets are networks of hijacked computer devices used to carry out various scams and cyberattacks. It is a large collection of compromised hosts that can be controlled by an attacker (Bot Master)
Χρειάζεται τα bot, να είναι συνδεδεμένα στο δίκτυο. Δεν έχει τεκμηριωθεί ότι είναι επιτυχές για να έχεις κάποιο κέρδος.

Botnets can be rent for all sorts of malicious activities. (Click fraud, SPAM, retweets, DDos)

How Botnet Works
Botnets are built to grow, automate, and speed up a hacker's ability to carry out larger attacks.

One person or even a small team of hackers can only carry out so many actions on their local devices. But, at little cost and a bit of time invested, they can acquire tons of additional machines to leverage for more efficient operations.

Bot master controls the Botnet through a hidden command and control channel.
-> bots periodically check this channel to receive new commands.

Network Scanning:
The purpose of network scanning is to manage, maintain, and secure the system using data found by the scanner. Network scanning is used to recognize available network services, discover, and recognize any filtering systems in place, look at what operating systems are in use, and to protect the network from attacks.

-> Interact with other hosts remotely to infer: (nmap = it's a tool showing στο δίκτυο)

Elia Nicolaou EPL326 Final Exam

- Operating system, based on slightly different implementations of network protocols.
- Running Services based on different ports.
- Versions of installed software, based on the application-layer replies.

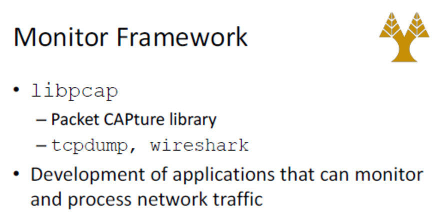
Network Monitoring:
Network monitoring is the process of monitoring the availability, uptime, operation, and performance of complex networks. This involves tracking and analyzing network components like routers, switches, and firewalls, and the connections between them.
οι αμυνόμενοι πρέπει να κοιτάζουν το δίκτυο για να προστατεύονται:
=> record and process network traffic
=> detect known attacks
=> detect anomalies
=> drop malicious traffic.

εισαγωγή κάποιου **monitor:** traffic analysis : όλη η κίνηση αντιγράφεται όποτε κάνουμε κάποιο traffic analysis.
Traffic analysis is the process of intercepting and examining messages in order to deduce information from patterns in communication, which can be performed even when the messages are encrypted.

There are some basic monitoring actions that u can take:

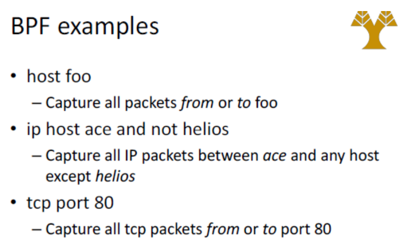
a. Firewalls :
A firewall is a system designed to prevent unauthorized access to or from a private network. You can implement a firewall in either hardware or software form, or a combination of both. Firewalls prevent unauthorized internet users from accessing private networks connected to the internet, especially intranets.
-> use a rule set with allowed services.
-> inspect some packet headers (so do not inspect the payload & it is relatively fast)
Μπορούμε να κάνουμε enforce κάποιους κανόνες. Είναι πολύ effective, για να κόβει την κίνηση όμως τους κανόνες τους αποφασίζουμε εμείς. Enforce rules, drop all ICMP packets with ECHO_REQUEST, drops pings. (ex)

Intrusion Detection System (IDS)
An Intrusion Detection System (IDS) is a system that monitors network traffic for suspicious activity and issues alerts when such activity is discovered. It is a software application that scans a network or a system for harmful activity or policy breaching.=> inspect the payload of the every packet and takes decisions
 based on the payloads.
 - Deep Packet Inspection (DPI)
 - slow, use of regular expressions.
=> complicated signatures.

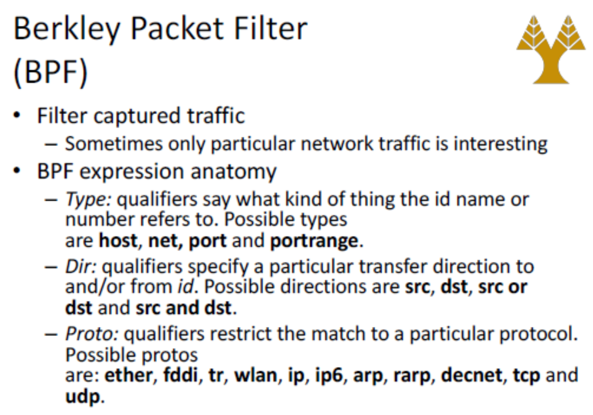

Monitor Framework

- `libpcap`
  - Packet CAPture library
  - `tcpdump, wireshark`
- Development of applications that can monitor and process network traffic

Berkley Packet Filter: (BPF)
Berkeley Packet Filters (BPF) provide a powerful tool for
 intrusion detection analysis. Use BPF filtering to quickly reduce large packet captures to a reduced set of results by filtering based on a specific type of traffic. Both admin and non-admin users can create BPF filters.
- Filter captured traffic: sometimes
only particular network traffic is interesting.


BPF examples

- host foo
  - Capture all packets *from* or *to* foo
- ip host ace and not helios
  - Capture all IP packets between *ace* and any host except *helios*
- tcp port 80
  - Capture all tcp packets *from* or *to* port 80


Berkley Packet Filter (BPF)

- Filter captured traffic
  - Sometimes only particular network traffic is interesting
- BPF expression anatomy
  - *Type:* qualifiers say what kind of thing the id name or number refers to. Possible types are **host**, **net**, **port** and **portrange**.
  - *Dir:* qualifiers specify a particular transfer direction to and/or from *id*. Possible directions are **src**, **dst**, **src or dst** and **src and dst**.
  - *Proto:* qualifiers restrict the match to a particular protocol. Possible protos are: **ether**, **fddi**, **tr**, **wlan**, **ip**, **ip6**, **arp**, **rarp**, **decnet**, **tcp** and **udp**.

Elia Nicolaou EPL326 Final Exam

Transport Layer Security (TLS)
Transport Layer Security, or TLS, is a widely adopted security protocol designed to facilitate privacy and data security for communications over the Internet. A primary use case of TLS is encrypting the communication between web applications and servers, such as web browsers loading a website.

* HTTPS is an implementation of TLS encryption on top of the HTTP protocol, which is used by all websites as well as some other web services. Any website that uses HTTPS is therefore employing TLS encryption.

http, imap,smtp,ssh,dsn => produce plain data, https uses tls.
Το encrypted part του TLS ξεκινά κάπου στην μέση του TCP handshake. Αν το TLS πάθει κάτι την πατήσαμε, όλες οι encrypted πληροφορίες μας θα μαθευτούνε.

* TLS allows applications to communicate over network using encryption. So we have sockets that send encrypted data.
*usually is supported in a different port & many applications supports TLS.

There are three main components to what the TLS protocol accomplishes: **Encryption, Authentication, and Integrity.**

1. Encryption: hides the data being transferred from third parties.
2. Authentication: ensures that the parties exchanging information are who they claim to be.
3. Integrity: verifies that the data has not been forged or tampered with.

But why we need to have TLS?
-> let us go back to what we learned, we have Alice and bob, and between them a lot of intermediate nodes.

the Man in the Middle:
* Intermediate nodes can be attacker controlled, routers, getaways, wireless access point and proxies.
a plain traffic is very easily to be compromised.
- monitored, leak passwords, credit card and more.
- modified (integrity problems) , change the contents of an email of a financial transaction.

History of TLS: created by Netscape (Netscape navigator => Firefox)
ωστόσο το πρωτόκολλο ακόμη δουλεύεται.

TLS Handshake: TLS handshake is the process that kicks off a communication session that uses TLS encryption. During a TLS handshake, the two communicating sides exchange messages to acknowledge each other, verify each other, establish the encryption algorithms they will use, and agree on session keys.

* several slightly different forms based on the cipher suite used.

What happens during a TLS handshake?
During the course of a TLS handshake, the client and server together will do the following:

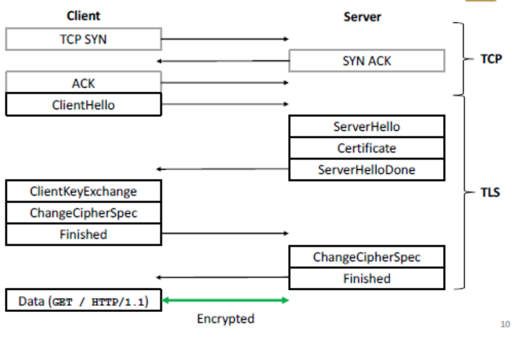
- Specify which version of TLS (TLS 1.0, 1.2, 1.3, etc.) they will use
- Decide on which cipher suites (see below) they will use
- Authenticate the identity of the server via the server's public key and the SSL certificate authority's digital signature
- Generate session keys in order to use symmetric encryption after the handshake is complete

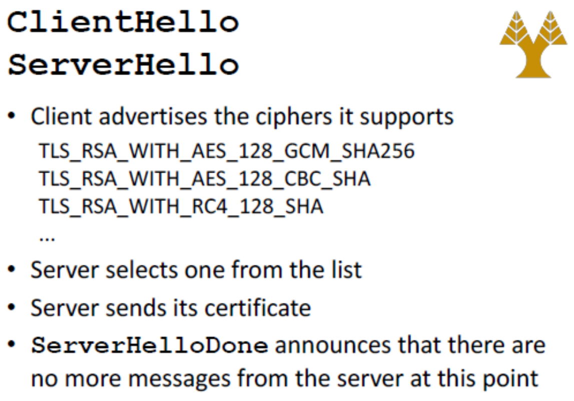
The handshake also handles authentication, **which usually consists of the server proving its identity to the client**. This **is done using public keys**. Public keys are encryption keys that **use one-way encryption**, meaning that anyone with the public key can unscramble the data encrypted with the server's private key to ensure its authenticity, but only the original sender can encrypt data with the private key. The server's public key is part of its TLS certificate.

Once data is encrypted and authenticated, it is then signed with a message authentication code (MAC). The recipient can then verify the MAC to ensure the integrity of the data. This is kind of like the tamper-proof foil found on a bottle of aspirin; the consumer knows no one has tampered with their medicine because the foil is intact when they purchase it.

εν μέρη authentication : πιστοποιεί το ένα άκρο , ο server δεν γνωρίζει και ακριβώς με ποιον μιλάει.

TLS Handshake





`ClientHello`
`ServerHello`

- Client advertises the ciphers it supports
  TLS_RSA_WITH_AES_128_GCM_SHA256
  TLS_RSA_WITH_AES_128_CBC_SHA
  TLS_RSA_WITH_RC4_128_SHA
  ...
- Server selects one from the list
- Server sends its certificate
- `ServerHelloDone` announces that there are no more messages from the server at this point

Cipher Suite:

The TLS handshake establishes a cipher suite for each communication session. The cipher suite is a set of algorithms that specifies details such as which shared encryption keys, or session keys, will be used for that particular session. TLS is able to set the matching session keys over an unencrypted channel thanks to a technology known as public key cryptography.

Client Key Exchange (Server → Client)

The Client Key Exchange message is sent right after the Server Hello Done is received from the server. If the server requests a Client Certificate, the Client Key Exchange is sent after that. During this stage, the client creates a pre-master key. The client encrypts a secret with the server’s public key.

Client Change Cipher Spec (Client → Server)

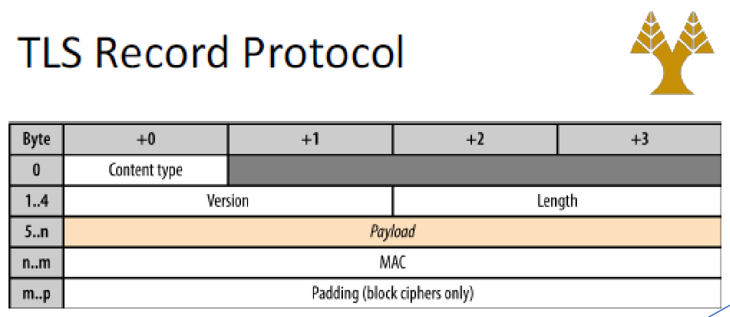
At this point, the client is ready to switch to a secure, encrypted environment. The Change Cipher Spec protocol is used to change the encryption. Any data sent by the client from now on will be encrypted using the symmetric shared key. From now on messages going to be encrypted.

Client Handshake Finished (Client → Server)

The last message of the handshake process from the client signifies that the handshake is finished. This is also the first encrypted message of the secure connection. The last message contains a MAC of all handshake messages exchanged.

Server Change Cipher Spec (Server → Client)

The server is also ready to switch to an encrypted environment. (finished). Any data sent by the server from now on will be encrypted using the symmetric shared key. The server decrypts the secret using it’s certificate’s private key and derives the master secret and communication keys.



TLS Record Protocol

Byte	+0	+1	+2	+3
0	Content type			
1..4	Version		Length	
5..n	*Payload*			
n..m	MAC			
m..p	Padding (block ciphers only)			

The right record size
- Small records have larger CPU overhead due to frequent MAC verification
- Large records will have to be reassembled by the TCP layer before they can be processed by the TLS layer
- Not always possible to tune the record size

TLS authentication:

For server authentication, the client uses the server's public key to encrypt the data that is used to compute the secret key. The server can generate the secret key only if it can decrypt that data with the correct private key.

For client authentication, the server uses the public key in the client certificate to decrypt the data the client sends during step 5 of the handshake. The exchange of finished messages that are encrypted with the secret key (steps 7 and 8 in the overview) confirms that authentication is complete.

If any of the authentication steps fail, the handshake fails and the session terminates.

During the TLS handshake the server sends a certificate to the client. The certificate is an electronic document, used to prove the ownership of a public key. The certificate includes important information about the key, the identity of its owner and the digital signature of an entity that has verified the certificate's contents.
If the signature is valid and the issuer is trusted, then the public kay is accepted.

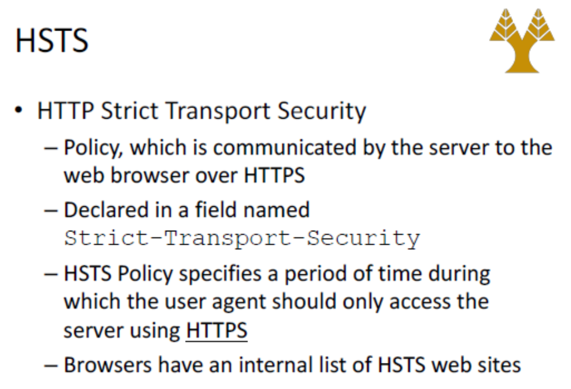
SSL Stripping:

SSL stripping is a technique by which a website is downgraded from https to http. In other words, the attack is used to circumvent the security which is enforced by SSL certificates on https sites. An MtM attacker can modify the server response and change all HTTPS links to point back to HTTP.

HSTS:

STS stands for HTTP Strict Transport Security, it's a web security policy mechanism that forces web browsers to interact with websites only via secure HTTPS connections (and never HTTP). This helps to prevent protocol downgrade attacks and cookie hijacking.

* απαγορεύει request σε http.



HSTS

- HTTP Strict Transport Security
  - Policy, which is communicated by the server to the web browser over HTTPS
  - Declared in a field named `Strict-Transport-Security`
  - HSTS Policy specifies a period of time during which the user agent should only access the server using HTTPS
  - Browsers have an internal list of HSTS web sites

Elia Nicolaou EPL326 Final Exam

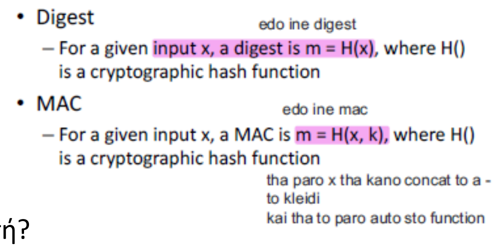
Message Authentication Codes (MAC)
* θα θέλαμε να φτιάξουμε μια hash function που θα παράγει digest που δεν μπορούν να τα παράγουν όλοι.

-> Digital Signing:
τι κάνει ουσιαστικά? συνδέει ένα hash function με ένα μη συμμετρικό κλειδί.
θέλουμε να δημιουργήσουμε μια παρόμοια procedure αλλά χωρίς να χρησιμοποιούμε το public key.
-one shared key
-cryptographic hash function (e.g SHA2)
-somehow, we have to mix the key with the hash function

MAC: a message authentication code (MAC), sometimes known as a tag, is a short piece of information used to authenticate a message—in other words, to confirm that the message came from the stated sender (its authenticity) and has not been changed.

Macs vs digest:

- A Message Digest is simply a hash of a message. It's the output of a cryptographic hash function applied to input data, which is referred to as a message.
- A Message Authentication Code (MAC) is a piece of information that proves the integrity of a message and cannot be counterfeited easily.



- Digest
  - For a given input x, a digest is m = H(x), where H() is a cryptographic hash function

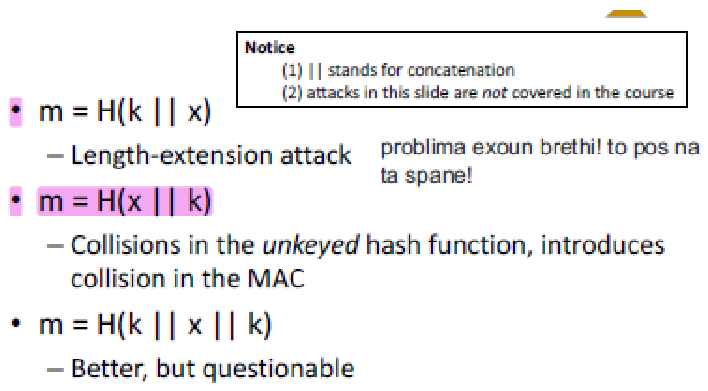
edo ine digest

- MAC
  - For a given input x, a MAC is m = H(x, k), where H() is a cryptographic hash function

edo ine mac

tha paro x tha kano concat to a -
to kleidi
kai tha to paro auto sto function

Καλά πως δημιουργώ την συνάρτηση αυτή?



Notice
(1) || stands for concatenation
(2) attacks in this slide are *not* covered in the course

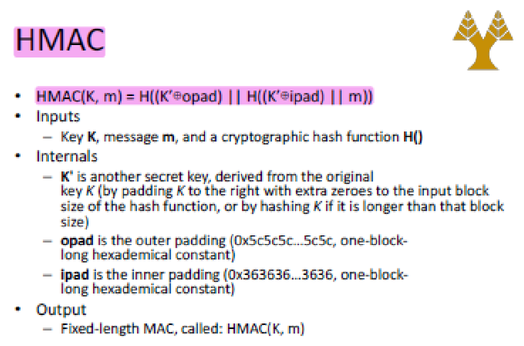
- m = H(k || x)
  - Length-extension attack

problima exoun brethi! to pos na ta spane!

- m = H(x || k)
  - Collisions in the *unkeyed* hash function, introduces collision in the MAC
- m = H(k || x || k)
  - Better, but questionable

το δεύτερο δεν μας στηρίζει , το πρώτο σπάει και το τρίτο δεν έχει αναλυθεί πολύ.

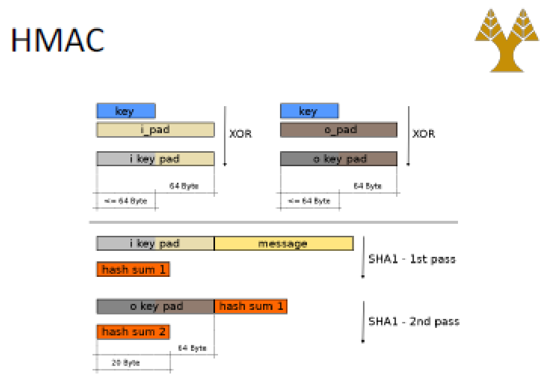
HMAC: In cryptography, an HMAC (sometimes expanded as either keyed-hash message authentication code or hash-based message authentication code) is a specific type of message authentication code (MAC) involving a cryptographic hash function and a secret cryptographic key.

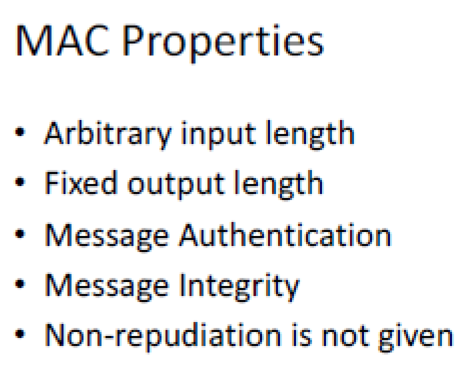


HMAC

- HMAC(K, m) = H((K'⊕opad) || H((K'⊕ipad) || m))
- Inputs
  - Key **K**, message **m**, and a cryptographic hash function **H()**
- Internals
  - **K'** is another secret key, derived from the original key *K* (by padding *K* to the right with extra zeroes to the input block size of the hash function, or by hashing *K* if it is longer than that block size)
  - **opad** is the outer padding (0x5c5c5c…5c5c, one-block-long hexademical constant)
  - **ipad** is the inner padding (0x363636…3636, one-block-long hexademical constant)
- Output
  - Fixed-length MAC, called: HMAC(K, m)

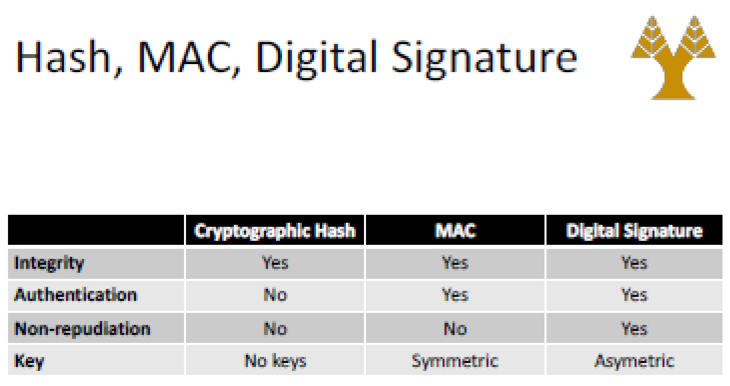
HMAC





MAC Properties

- Arbitrary input length
- Fixed output length
- Message Authentication
- Message Integrity
- Non-repudiation is not given



Hash, MAC, Digital Signature

	Cryptographic Hash	MAC	Digital Signature
Integrity	Yes	Yes	Yes
Authentication	No	Yes	Yes
Non-repudiation	No	No	Yes
Key	No keys	Symmetric	Asymetric

Elia Nicolaou EPL326 Final Exam

Introduction to Web Security:

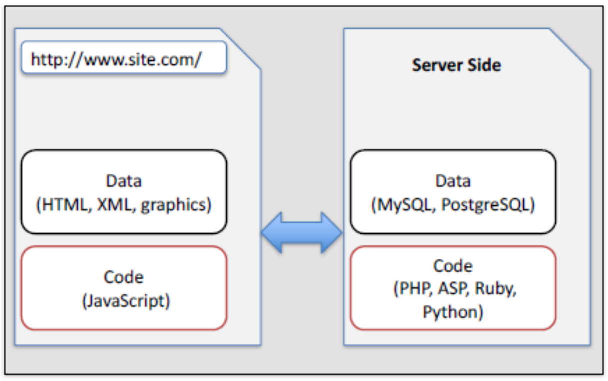
Web Applications : τα web application αποτελούν ένα νέο πρόβλημα.
Τι είναι καν? A web application is application software that runs on a web server, unlike computer-based software programs that are run locally on the operating system of the device.

2 parts
1. code executing in the browser environment (might be javascript / it's the client part or client side)
2. Code executing in a web server
   PHP,Ruby,Python

*client side communicates with the server side using the HTTP protocol.

Αριστερά
Client Side

Δεξιά
Server Side



Interactive Web Apps:
Interactive Web Application refers to web application which has a lot interactivity to it, i.e. elements that can receive user inputs and reacts accordingly.
τα request δεν συσχετίζονται πλέον είναι interactive -> δίνει την δυνατότητα του να αλλάξεις τα δεδομένα δυναμικά.

Web 2.0 is the term used to describe a variety of web sites and applications that allow anyone to create and share online information or material they have created. A key element of the technology is that it allows people to create, share, collaborate & communicate.

Same origin policy:
The same-origin policy is a critical security mechanism that restricts how a document or script loaded from one origin can interact with a resource from another origin. It helps isolate potentially malicious documents, reducing possible attack vectors.

as example : the scripts allowed to run have the same origin. (scheme, hostname, and port number)

Cross-Site Scripting (XSS)
Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.
An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page.

Elia Nicolaou EPL326 Final Exam

Reflective XSS:

Reflected attacks are those where the injected script is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request. Reflected attacks are delivered to victims via another route, such as in an e-mail message, or on some other website. When a user is tricked into clicking on a malicious link, submitting a specially crafted form, or even just browsing to a malicious site, the injected code travels to the vulnerable web site, which reflects the attack back to the user's browser. The browser then executes the code because it came from a "trusted" server. Reflected XSS is also sometimes referred to as Non-Persistent or Type-II XSS.

XSS solutions :

1. Implement Content Security Policy (CSP)

There is another good complex solution to mitigate the impact of an XSS flaw called Content Security Policy. It's a browser side mechanism which allows you to create source allow lists for client side resources of your web application, e.g. JavaScript, CSS, images, etc. CSP via special HTTP header instructs the browser to only execute or render resources from those sources.

2. Filter our data/code (although is very hard to implement)

Cross-Site Request Forgery:

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

CSRF prevention:

- Check if your framework has built-in CSRF protection and use it
  If framework does not have built-in CSRF protection add CSRF tokens to all state changing requests (requests that cause actions on the site) and validate them on backend
- Always use SameSite Cookie Attribute for session cookies
- Implement at least one mitigation from Defense in Depth Mitigations section
- Use custom request headers
- Verify the origin with standard headers
- Use double submit cookies
- Consider implementing user interaction based protection for highly sensitive operations
- Remember that any Cross-Site Scripting (XSS) can be used to defeat all CSRF mitigation techniques!
- See the OWASP XSS Prevention Cheat Sheet for detailed guidance on how to prevent XSS flaws.
- Do not use GET requests for state changing operations.
- If for any reason you do it, you have to also protect those resources against CSRF

-> check referrer, check origin header, sensitive forms include a hidden random token.

Elia Nicolaou EPL326 Final Exam

SQL injection and other Web Attacks:

Sql injection :

SQL injection usually occurs when you ask a user for input, like their username/userid, and instead of a name/id, the user gives you an SQL statement that you will unknowingly run on your database
όλες σχεδόν οι εφαρμογές χρησιμοποιούν μια βάση δεδομένων, έτσι καλή ιδέα είναι να κάνει κάποια επίθεση με την χρήση της γλώσσας.
=> that happens because web apps often use persistent storage – database

το πρόβλημα θα προκύψει όταν το input εξαρτάται από τον χρήστη, ο οποίος ναι μπορεί να δώσει κάτι malicious.

SQL queries and web apps:
-> web apps construct SQL queries dynamically.

```
sql_query = " SELECT * FROM users WHERE
       name = ' " + user_name + " '; ″
db.execute_query(sql_query);
```

το σημείο που δίνει ο χρήστης είναι attacker controlled.
point is to add another ' and add with ; your query.

ποια είναι η ιδέα της εύκολης άμυνας? να κοιτάζουμε τα μονά εισαγωγικά, φτιάχνεις το query μέσα από ένα template, ένα API

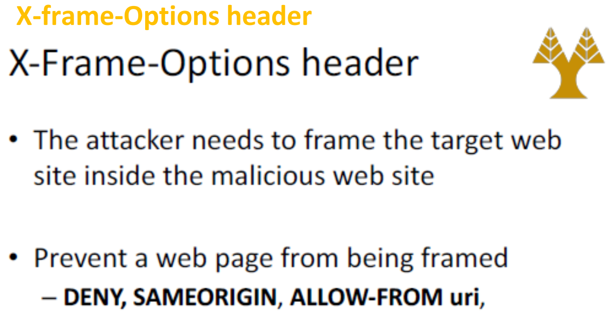
prepared statements using SQL templates

```
stmt = db.prepare("SELECT * FROM users WHERE
user_name= ?");
stmt.set(1, user_name);
stmt.execute();
```

Clickjacking (web attacks)

Clickjacking is an attack that tricks a user into clicking a webpage element which is invisible or disguised as another element. This can cause users to unwittingly download malware, visit malicious web pages, provide credentials or sensitive information, transfer money, or purchase products online.

X-frame-Options header



X-Frame-Options header

- The attacker needs to frame the target web site inside the malicious web site
- Prevent a web page from being framed
  – **DENY, SAMEORIGIN, ALLOW-FROM uri,**

*automated data input : scripts can fill in web forms (create automatically web forms) -> CAPTCHAS is the solution.

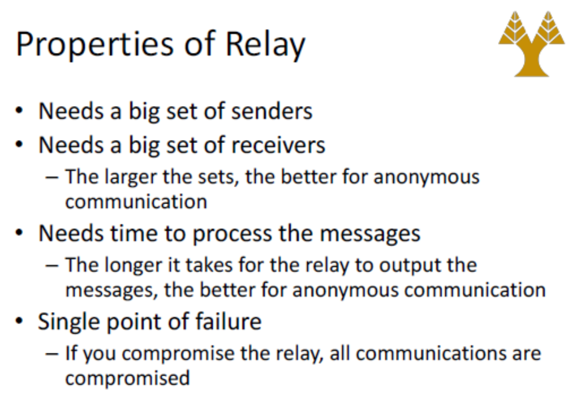
Elia Nicolaou EPL326 Final Exam

The onion router (TOR)

Ο Oscar ωστόσο παρόλο που μπορεί να μεν ξέρει τα ενδιάμεσα, ξέρει τι βρίσκετε στα άκρα. Μπορεί να δει τα endpoints.
άρα **προκύπτει νέο πρόβλημα :** Το πρόβλημα της ανωνυμίας (δεν πρέπει να γνωρίζει τα άκρα).

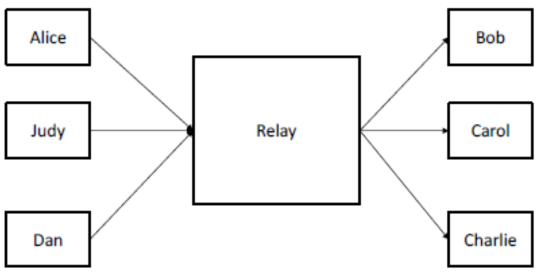
Anonymous Communication:
the purpose is for users to interact with other users over the internet but without the MitM should not be able to infer who is talking to whom.

<u>πώς θα καταφέρουμε να πετύχουμε το anonymous communication?</u> => θα πάμε να κρυφτούμε πίσω από το proxy.



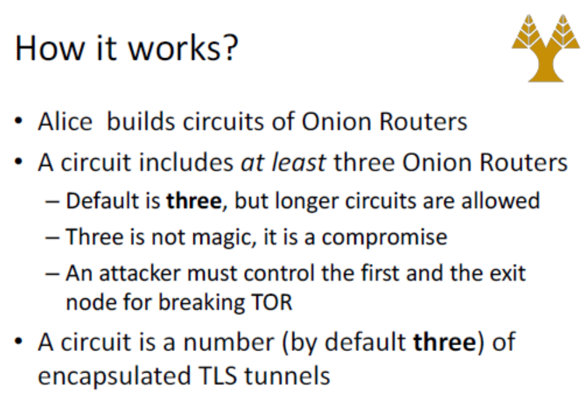
Properties of Relay

- Needs a big set of senders
- Needs a big set of receivers
  - The larger the sets, the better for anonymous communication
- Needs time to process the messages
  - The longer it takes for the relay to output the messages, the better for anonymous communication
- Single point of failure
  - If you compromise the relay, all communications are compromised



Goals:
*we need anonymity for practical low-latency communications (fast like we browsing)
* defending a realistic threat model (o attacker den einai teleios adinamos)
- attacker cannot monitor all internet links
-can observe some fraction of the network traffic
-can generate,modify,delete or delay traffic.
-can operate onion routers of their own
-can compromise some fraction of the onion routers.

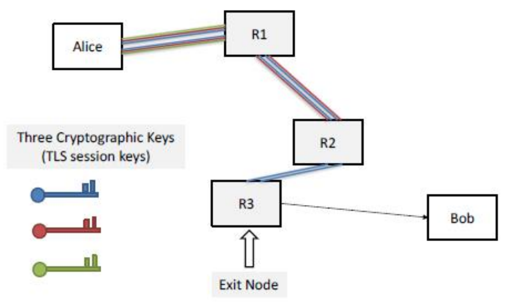
The onion router (TOR)

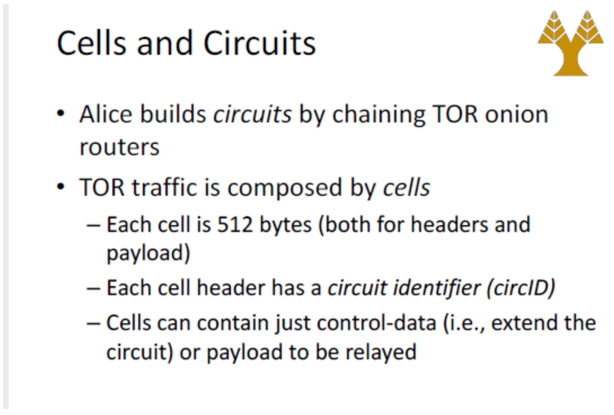


How it works?

- Alice builds circuits of Onion Routers
- A circuit includes *at least* three Onion Routers
  - Default is **three**, but longer circuits are allowed
  - Three is not magic, it is a compromise
  - An attacker must control the first and the exit node for breaking TOR
- A circuit is a number (by default **three**) of encapsulated TLS tunnels

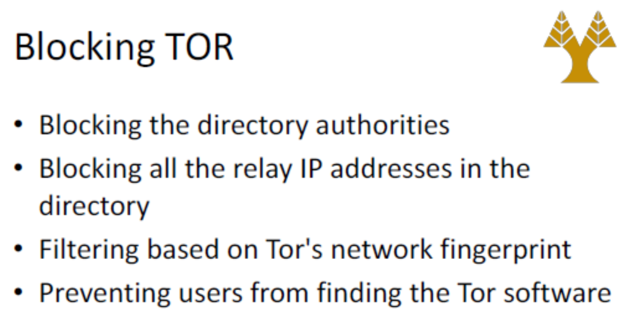
Onion Routing





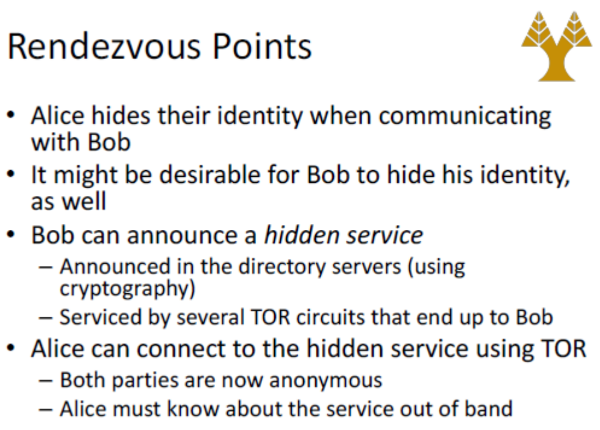
Cells and Circuits

- Alice builds *circuits* by chaining TOR onion routers
- TOR traffic is composed by *cells*
  - Each cell is 512 bytes (both for headers and payload)
  - Each cell header has a *circuit identifier (circID)*
  - Cells can contain just control-data (i.e., extend the circuit) or payload to be relayed



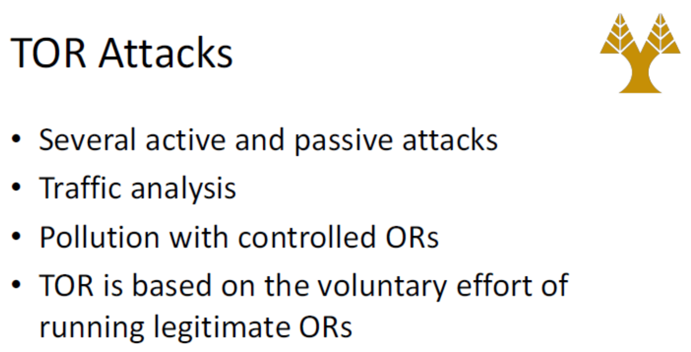
Blocking TOR

- Blocking the directory authorities
- Blocking all the relay IP addresses in the directory
- Filtering based on Tor's network fingerprint
- Preventing users from finding the Tor software



Rendezvous Points

- Alice hides their identity when communicating with Bob
- It might be desirable for Bob to hide his identity, as well
- Bob can announce a *hidden service*
  - Announced in the directory servers (using cryptography)
  - Serviced by several TOR circuits that end up to Bob
- Alice can connect to the hidden service using TOR
  - Both parties are now anonymous
  - Alice must know about the service out of band



TOR Attacks

- Several active and passive attacks
- Traffic analysis
- Pollution with controlled ORs
- TOR is based on the voluntary effort of running legitimate ORs