Computer Security

04r. Pre-exam 1 Concept Review

Paul Krzyzanowski
TAs: Fan Zhang, Shuo Zhang
Rutgers University
Fall 2019

Assignment 4 Review

Question 1

What is meant by the term **shellcode** when constructing exploits?

It is "a small piece of code used as the payload in the exploitation of a software vulnerability.

"It is called "shellcode" because it typically starts a command shell from which the attacker can control the compromised machine."

Question 2

What is a **NOP sled** (also known as a **landing pad**)?

A sequence of NOP (no-operation) instructions that are used when the attacker does not know the precise location of the starting address of the injected shellcode.

The return address can be set to any value within this stream of NOP instructions and the processor will skip through them until it gets to useful code.

Question 3

What is a **gadget**?

In Return-Oriented Programming (ROP), gadgets are a small group of instructions ending with a *return* (RET) instruction.

These instructions are already present in the program or libraries used by the program so they do not need to be injected.

All that needs to be injected is a set of new stack frames containing return addresses that will cause program execution to go from one gadget to another to another.

Question 3

What four techniques are presented to mitigate **SQL injection attacks**?

1. Parameterized statements (prepared statements or stored procedures)
   All these methods separate the query from user input. User-supplied parameters never become part of the query.
2. Escaping special characters
   This disables characters that might otherwise mark the end of a string or the start of a new query
3. Pattern checking
   Ensure that all requested fields contain expected values. Example: if you expect a user name, you might check that you have only sequence of alphanumeric characters.
4. Database permissions
   Restrict operations on or access to certain tables
5. Detection
   Use filters similar to firewalls or spam filters. Match patterns and disable access from hosts that send a sequence of bad queries (they may be experimenting, trying to find an attack that works).

Key ideas from the past four lectures

Computer security

- What computer security addresses:
  - **Confidentiality**
    - Allow only authorized users to access data & resources
    - Privacy: limit what information will be shared with others
    - Privacy is a reason for confidentiality
  - **Integrity**: trustworthiness of data & resources
    - Data integrity: data hasn't been corrupted
    - Origin integrity/destination integrity: validate who is sending and who is receiving
    - System integrity: system works properly and has not been subverted
  - **Availability**
    - The system is available for use and performs properly

No easy answers

- Security is hard
  - Software is incredibly complex
  - Systems are complex: cloud + local; 3rd party components; multiple admins
- If it was easy, we wouldn't have massive security breaches year after year
  - No magic solutions

Security goals

- Prevention: prevent attackers from violating security policy
  - Implement mechanisms that users cannot override
  - *Example: ask for a password*
- Detection: detect & report attacks
  - Important when prevention fails
  - Indicates & identifies weaknesses with prevention
  - Also: detect attacks even if prevention is successful
- Recovery: stop the attack, repair damage
  - … Or continue to function correctly even if attack succeeds
  - Forensics: identify what happened so you can fix it
  - *Example: restoration from backups*

Policies & Mechanisms

Policy: description of what is or is not allowed
- E.g., users must have a password

Mechanisms: implement and enforce policies
- E.g., password entry & authentication

Definitions

- Vulnerability
  - A weakness in the implementation or operation of a system
  - Bugs, bad configuration, lack of access controls
- Attack
  - A means of exploiting a vulnerability
  - E.g., buffer overflow, social engineering
- Threat
  - An adversary that is capable of attacking
- Trusted Computing Base (TCB)
  - All hardware & software of a computing system critical to its security
    - Example: operating system & system software
    - If the TCB is compromised, you have no assurance that any aspect of the system is secure

Threat categories

- Disclosure: Unauthorized access to data
  - *Snooping (wiretapping)*
- Deception: Acceptance of false data
  - *Injection of data, modification of data, denial of receipt*
- Disruption: Interruption or prevention of correct operation
  - *Modification of the system, denial of service, delays*
- Usurpation: Unauthorized control of some part of a system
  - *Modification, spoofing an identity, escalation of privileges*

Access Control

Protection & Access Control

Protection

- The mechanism that provides and enforces controlled access of resources to processes
- A protection mechanism *enforces* security policies

Access control

- Ensure that authorized users can do what they are permitted to do … *and no more*

The Operating System

- Protect the OS from applications
- Make sure it stays in control
- Basic OS mechanisms
  - Hardware timer – periodically gives control to the OS
  - Scheduler – decides which process gets to run
  - Memory Management Unit (MMU) – provides private memory spaces and memory protection (read/write/execute access)
  - User & kernel mode execution – only the kernel can access privileged instructions

Access control: subjects & objects

- Subject: the thing that needs to access resources
  - Often the user
- Object: the resource the subject may access
- Access control: defines how subjects may access objects

Unix (POSIX) access control

- Each object (file, device) has
  - One owner and one group
  - Read, write, and/or execute permissions for the owner, group, and other (everyone else)
- Each subject (user) has
  - One user ID
  - Membership in one or more groups
- For directories
  - Execute permission = search permission
  - Write access = you can create/delete files or directories within that directory

POSIX file operations

- chmod: set file permissions
- chown: change file ownership of a file
- chgrp: change group ownership of a file
- Programs run with the permissions of the user who runs the program
- setuid: permission bit that causes an executable file to run with the ID of the file owner, not the user who is executing the file
  - WARNING! Many set UID programs run as root (administrator) and are attractive targets. If you can take control of that program then you get administrative privileges

Principle of least privilege

- Principle of least privilege
  - At each abstraction layer, every element (user, process, function) should be able to access ***only*** the resources necessary to perform its task
- Privilege separation
  - Divide a program into multiple parts: high & low privilege components

Access control matrix

- Table defining what a subject (user) can do to an object (file)
- Access control lists: store permissions with an object
- Capability lists: store permissions with a subject

objects *objects*

domains of protection (subjects)

	F_0	F_1	Printer
D_0	read	read-write	print
D_1	read-write-execute	read	
D_2	read-execute		
D_3		read	print
D_4			print

Access control list *Capability list*

DAC vs. MAC

- DAC = Discretionary Access Control
  - Users get to set access permissions
- MAC = Mandatory Access Control
  - Administrators set access permissions that users cannot overwrite

Multi-Level Security Models

- The **Bell-LaPadula** model is all about confidentiality
  - Simple security property:
    - You cannot read data from higher clearance levels than you are
  - Star *-property:
    - You cannot create data that is a lower clearance level than you are
  - Discretionary security property
    - Users can control access with ACLs only *after* MAC is enforced
- The **Biba** model is similar but is all about integrity
  - Simple integrity property:
    - You cannot read an object from a lower integrity level than you are
    - *Example: A process will not read a system configuration file created by a lower-integrity-level process*
  - Star *-property:
    - You cannot write to an object of a higher integrity level than you are
    - *Example: A web browser may not write a system configuration file*

Other MAC models

- **Type Enforcement (TE) Model**
  - An access control matrix that gets checked first
    - This is managed by an administrator
  - Subjects assigned to domains; objects assigned to types
  - Matrix defines domain-domain and domain-type transitions
- **Role-Based Access Control (RBAC) model**
  - Users are assigned roles (job functions)
  - Access permissions are granted to roles
  - Access rights have a *session:* you get them to do a task
  - Commonly used in database systems
    - Roles: *delete users, modify a user's pay, view users, …*

Multilateral Security

- In addition to levels, a level may have compartments
  - You can only access resources if you have been granted access to that compartment
  - E.g., {Top Secret, Elvis}
    - can access {Top Secret}, {Secret, Elvis}, {Secret}
    - Cannot access {Top Secret, UFO}, {Secret, UFO}
- Lattice model
  - Implements multilevel security with labels per level
  - Directed graph that defines access rights among clearance levels and compartment labels

Chinese Wall Model

- Defines conflict classes: groups of competing companies
  - Designed for businesses where employees have to avoid conflict of interest
- Basic rule
  - A subject can access objects from a company as long as it never accessed objects from competing companies.

Program Hijacking

Stack-based buffer overflow

- Buffer limits not checked
  - Often because unsafe functions like strcpy, strcat, and sprintf are used
- Overflow overwrites frame pointer & stack pointer
- If the stack pointer is changed, the return address is changed
  - Write code into the buffer
  - Overflow the buffer to set the return address
  - When the function returns, it branches to the new code

Off-by-one Buffer Overflows

An off-by-one stack overflow can only modify one byte of the top of the stack, which holds the frame pointer

- When a function returns, the modified frame pointer becomes the reference point for all local variables
- It also becomes the new stack pointer when a new function is called
- (see homework assignment)

Heap & text segment overflows

- A buffer overflow can overwrite adjacent variables that are allocated in higher memory
  - The program will use these modified variables

Printf format attacks

If an attacker can change the printf format string

- Read the stack
  - Read any address on the stack (using %x, for example)
  - If you don't supply arguments, printf will match %x with the next item on the stack
- Modify memory
  - Use "%x" to set where we write in memory: each %x skips one word on the stack
  - Use "%.Nx" to generate N bytes of output – this allows you to set the value you will write
  - Use %n to write the value – it prints the # of bytes output so far

Defenses

- Data Execute Protection (DEP)
  - Operating system turns off execute permission for stack and heap memory
  - Attacks:
    - *return-to-libc*: overflow a return address to a desired point in the C library
    - *Return-Oriented-Programming (ROP):* overflow a stack of return addresses to various points in libraries or the program – the return from one function takes you to the next entry point
- Address Space Layout Randomization (ASLR)
  - Load programs and libraries into different memory locations so addresses are different each time
- Stack Canaries
  - Compiler places a random # on the top of the stack and checks it before returning from a function

SQL Injection Attacks

- If user input becomes part of a SQL query, it can change the type of query – or add additional commands

```
SELECT * from logininfo WHERE username = paul AND password = 'abcde'

SELECT * from logininfo WHERE username = paul AND password = '' OR
1=1 -- ;'
```

  - Validate all input!
  - Safest prevention = use parameterized queries – don't make user input part of the command

Shell injection attacks

- Use of *system()* and *popen()* in programs
  - These invoke the shell. Same risk as SQL injection if user input is part of the command
- PATH variable: change the order in which the shell looks for programs
- LD_PRELOAD: preload libraries, possibly overriding functions that the program uses with your own
- LD_LIBRARY_PATH: similar attack – tell the OS where to look for libraries

App-level name parsing

- Parsing pathnames to make sure a user-supplied name stays within a subdirectory can be trickly
  - `http://poopybrain.com/../../../etc/passwd`
- Escaped Unicode characters make it harder
  - Single-byte characters have multi-byte equivalents: "/" = 0x2f = 0xc0af

TOCTTOU Attack

- Time Of Check To Time Of Use
  - If you check the condition and then do something, you may introduce a race condition
  - An attacker may change something after you check the condition but before you do the operation
    - Example: change a link to a user-readable file to a privileged file

App confinement

- chroot: change root directory for a process & its children
  - If an attacker becomes root, he may be able to escape by creating a device file that gives access to the disk or to memory
- FreeBSD Jails
  - Same namespace protection like chroot
  - But you can take power away from root for processes in the jail
    - No ability to create devices, raw sockets, mounting filesystems
  - Way more secure

App confinement

- Linux namespaces
  - Provide a private namespace for directory structure, network, process ID, user/group IDs, IPC, hostname
- Linux capabilities
  - Selectively take away power if a process becomes root.
  - Disallow file owner changes, permission changes, sending signals, creating raw sockets, changing root, etc.
- Linux control groups
  - Limit how much resources a process can use (CPU, memory, files, network)

The end