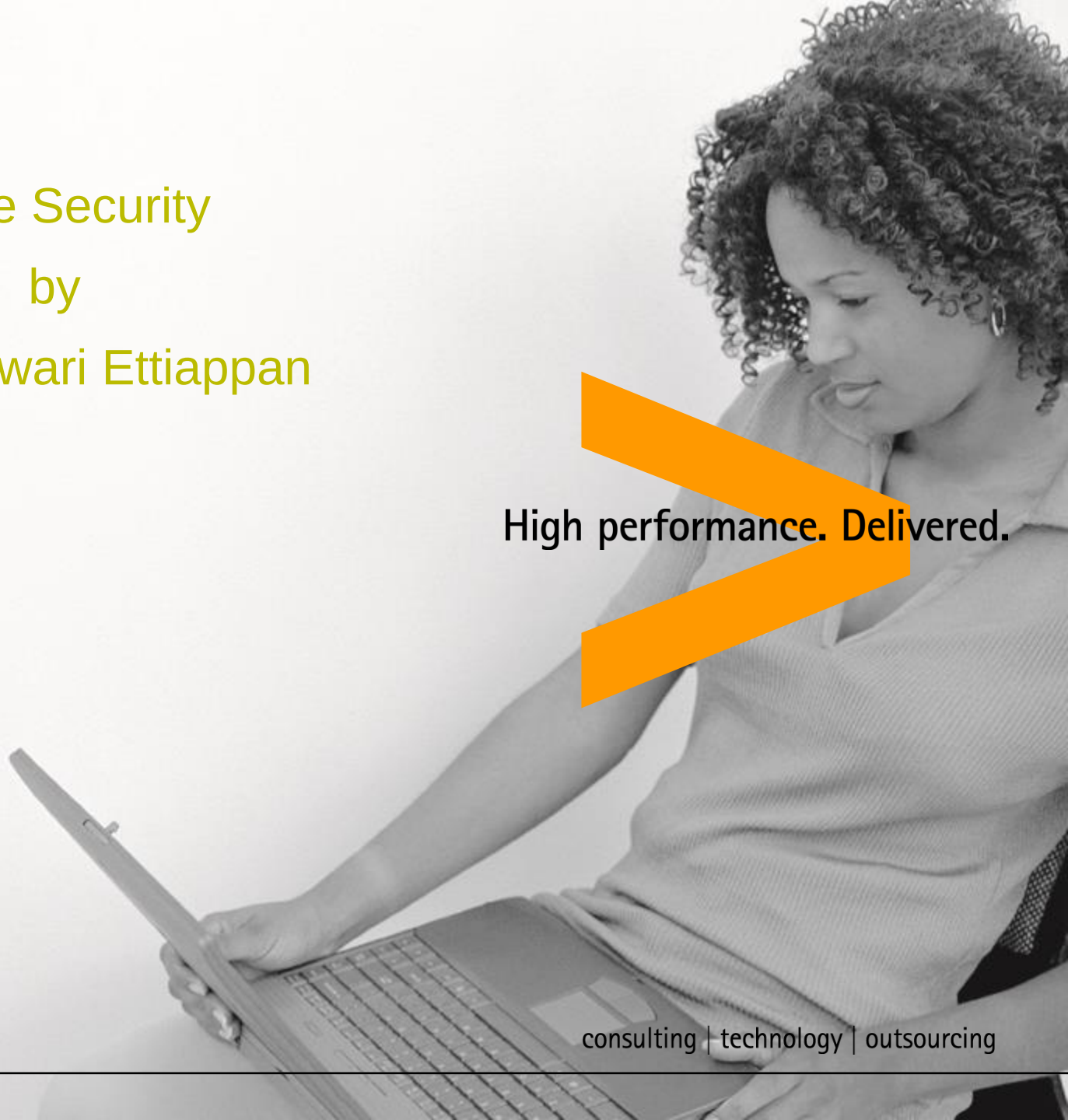
Code Security

by

Parameswari Ettiappan



High performance. Delivered.

consulting | technology | outsourcing

Goals

- Introduction to Java Security
- Secure Software Development
- File Input and Output and Serialization
- Input Validation
- Error Handling and Logging
- Authentication and Authorization
- Java Authentication and Authorization Service (JAAS)
- Java Concurrency and Session Management

Goals

- Java Cryptography
- Java Application Vulnerabilities
- Application-Level Cryptography
- Secure Development Practices

Goals

- User credentials are sent in clear / easy to decode format
- Apply Salted Hash technique on the password.
- Session Fixation
- Follow a secure session management lifecycle which includes proper initialization, maintenance, authentication and termination of the session token.
- Host Header Attack
- The web application should use the SERVER_NAME instead of the Host header.

Goals

- HTML Injection
  - Implement validations to escape all html tags and special characters from the user supplied input.
- Reflective Cross Site Scripting
  - The best way to prevent cross-site scripting is to make sure that the web application does not make use of user input in the subsequent HTML pages, without validating or sanitizing it .

Goals

- Authorization Bypass
  - Maintain proper authorization schema.
- Improper error handling
  - Create a default error handler which returns an appropriately sanitized error message for most users in production for all error paths.
- Vulnerable Remember Password
  - Ensure that no credentials are stored in clear text or are easily retrievable in encoded or encrypted forms in cookies.

Vulnerability

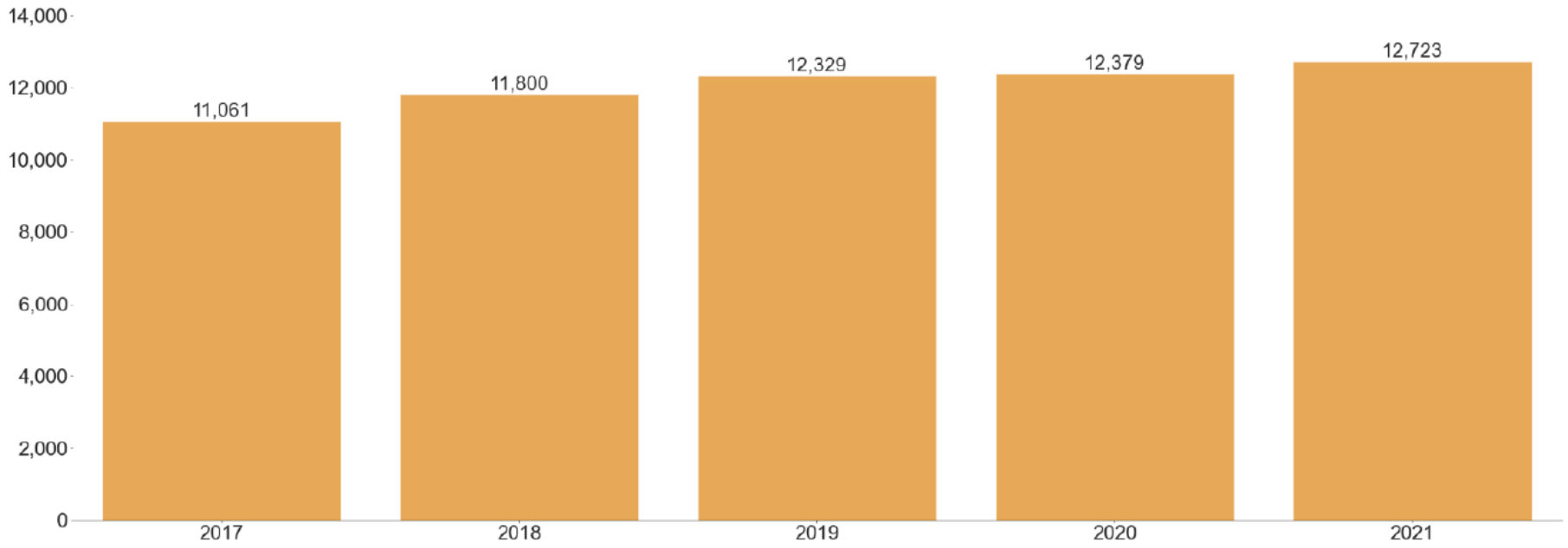
Sharks Are Scary but Worry About Mosquitoes

Vulnerability

Vulnerability Trends in 2021

2021 At A Glance

Figure 1: Number of vulnerabilities disclosed by Q2, in the last five years



Vulnerability

“Top” Products by Confirmed Vulnerabilities

Table 1: Top ten products by vulnerability disclosures in Q2 2021, as compared to 2020.

Name	Rank 2021	Rank 2020	Count 2021	Count 2020
Debian Linux	1 ↑	2	628	609
Fedora	2	N/A	584	N/A
openSUSE Leap	3 ↓	1	526	692
Ubuntu	4	4	443	521
Windows 10	5	5	274	478
SuSE Linux Enterprise Server (SLES)	6 ↑	10	260	394
Windows Server (Semi-Annual Channel)	7 ↑	8	259	427
Windows Server 2019	8 ↓	7	248	436
Google Pixel / Nexus Devices	9	9	242	414
SuSE Linux Enterprise Server for SAP	10 ↑	15	233	360

Vulnerability

“Top” Vendors by Confirmed Vulnerabilities

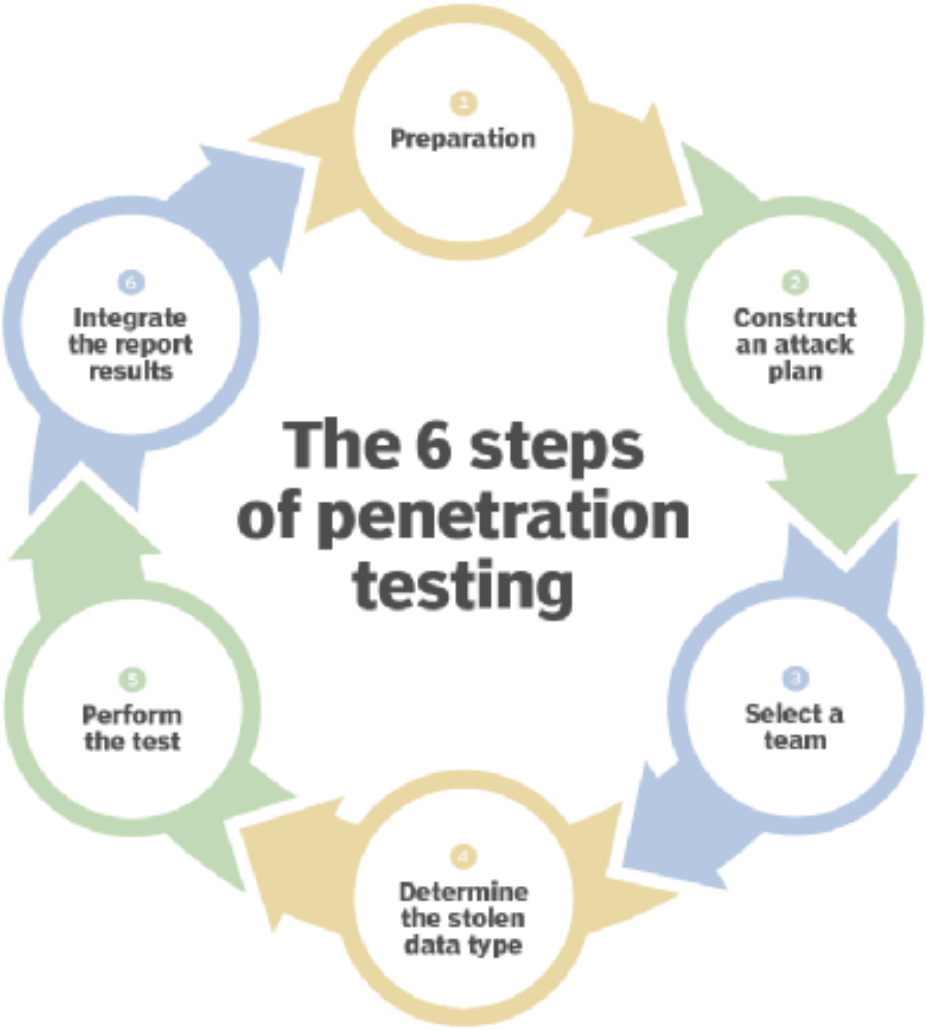
Table 2: Top ten vendors by vulnerability disclosures in Q2 2021, as compared to 2020

Name	Rank 2021	Rank 2020	Count 2021	Count 2020
Software in the Public Interest, Inc.	1 ↑	8	628	610
Microsoft Corporation	2 ↑	3	627	789
SUSE	3 ↑	4	590	782
Fedora Project	4	N/A	584	N/A
IBM Corporation	5 ↑	6	547	708
Oracle Corporation	6 ↓	1	521	915
Google	7 ↓	5	503	753
Cisco Systems	8 ↑	10	463	384
Canonical Ltd.	9	9	444	522
Red Hat	10 ↓	2	439	843

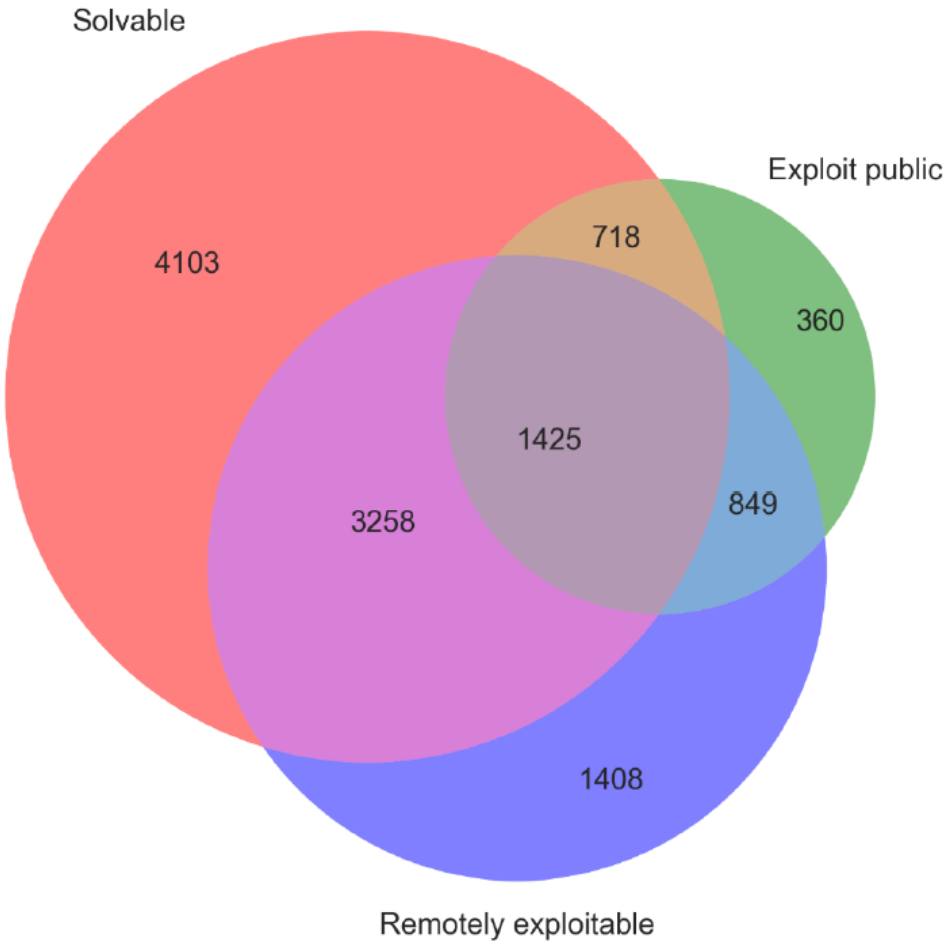
Vulnerability Disclosure Growth

- Vulnerability disclosure is the practice of reporting security flaws in computer software or hardware.
- Security researchers, IT security teams, in-house developers, third-party developers and others who work with the vulnerable systems may disclose vulnerabilities directly to the parties responsible for the flawed systems.
- Ensuring that software or hardware vendors can address vulnerabilities before bad actors can find and exploit them is crucial.
- Identifying such flaws is so important that bug bounties, or vulnerability rewards programs, which reward researchers for finding flaws, are often initiated along with internal code audits **and penetration tests** as part of an organization's vulnerability management strategy.

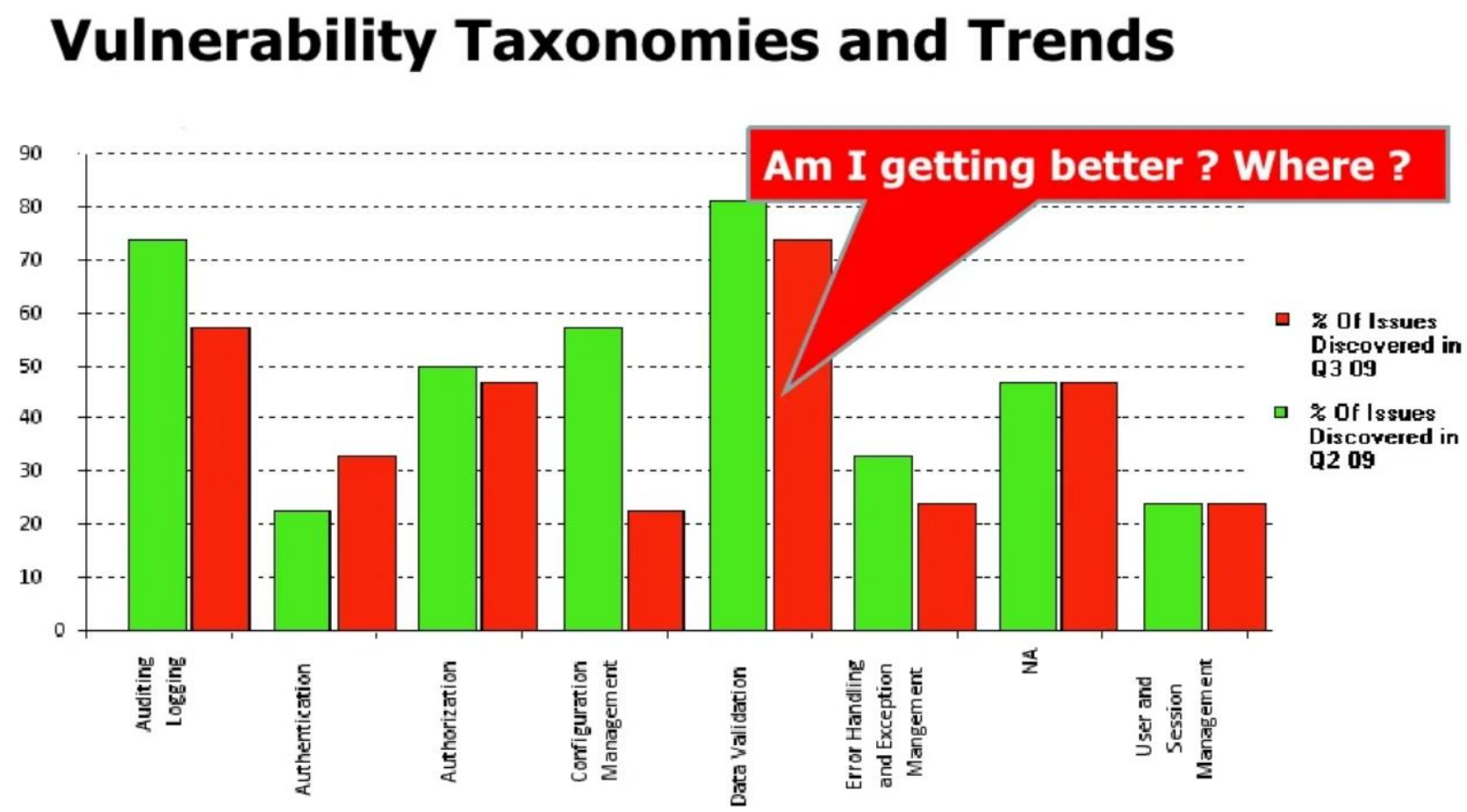
Penetration Testing



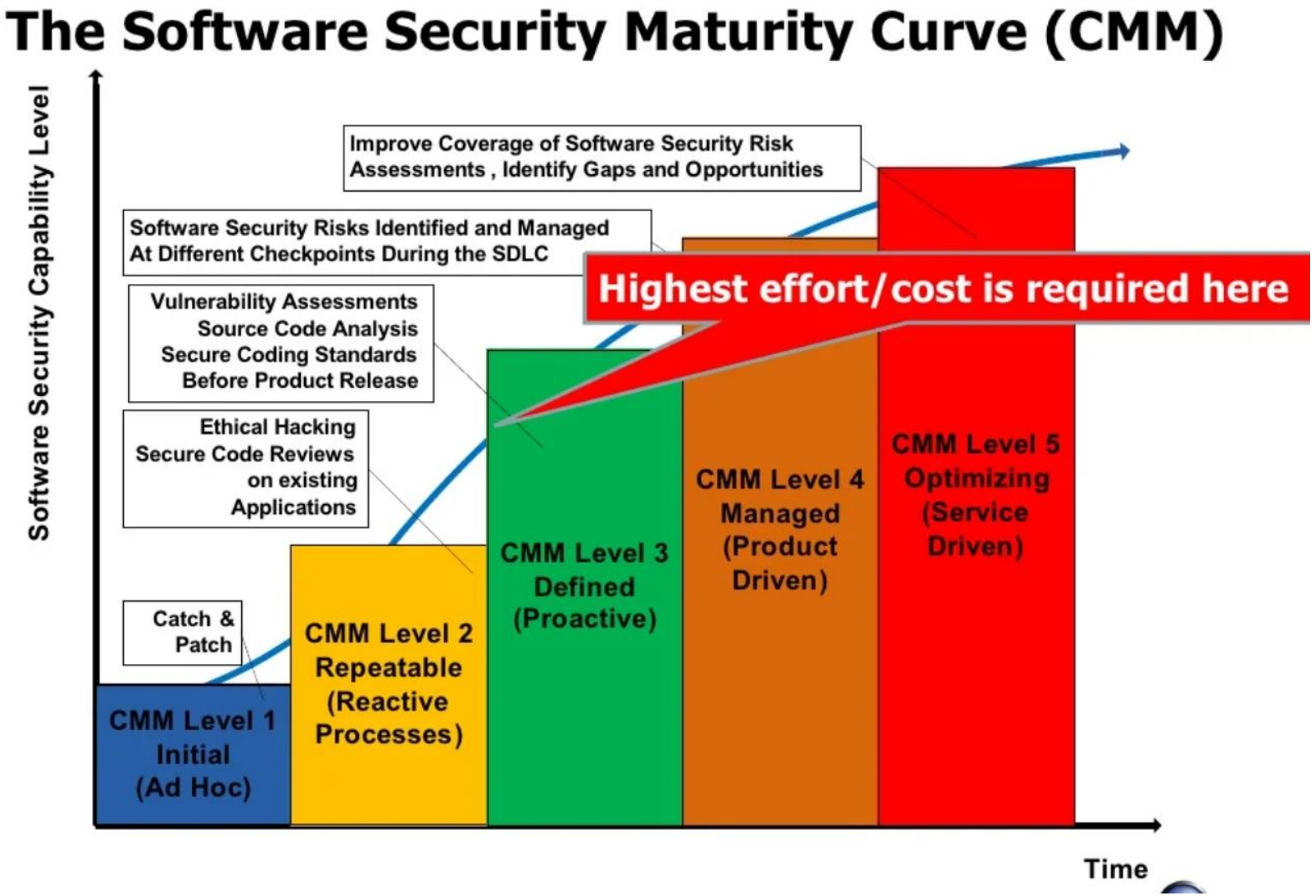
Importance of proper vulnerability intelligence



Vulnerability



Vulnerability



Vulnerability Scanning Tools

Tools Listing

Name/Link	Owner	License	Platforms	Note
Abbey Scan	MisterScanner	Commercial	SaaS	
Acunetix	Acunetix	Commercial	Windows, Linux, MacOS	Free (Limited Capability)
APIsec	APIsec	Commercial	SaaS	Free limited API Pen Test
App Scanner	Trustwave	Commercial	Windows	
AppCheck Ltd.	AppCheck Ltd.	Commercial	SaaS	Free trial scan available
AppScan	HCL Software	Commercial	Windows	

Vulnerability Scanning Tools

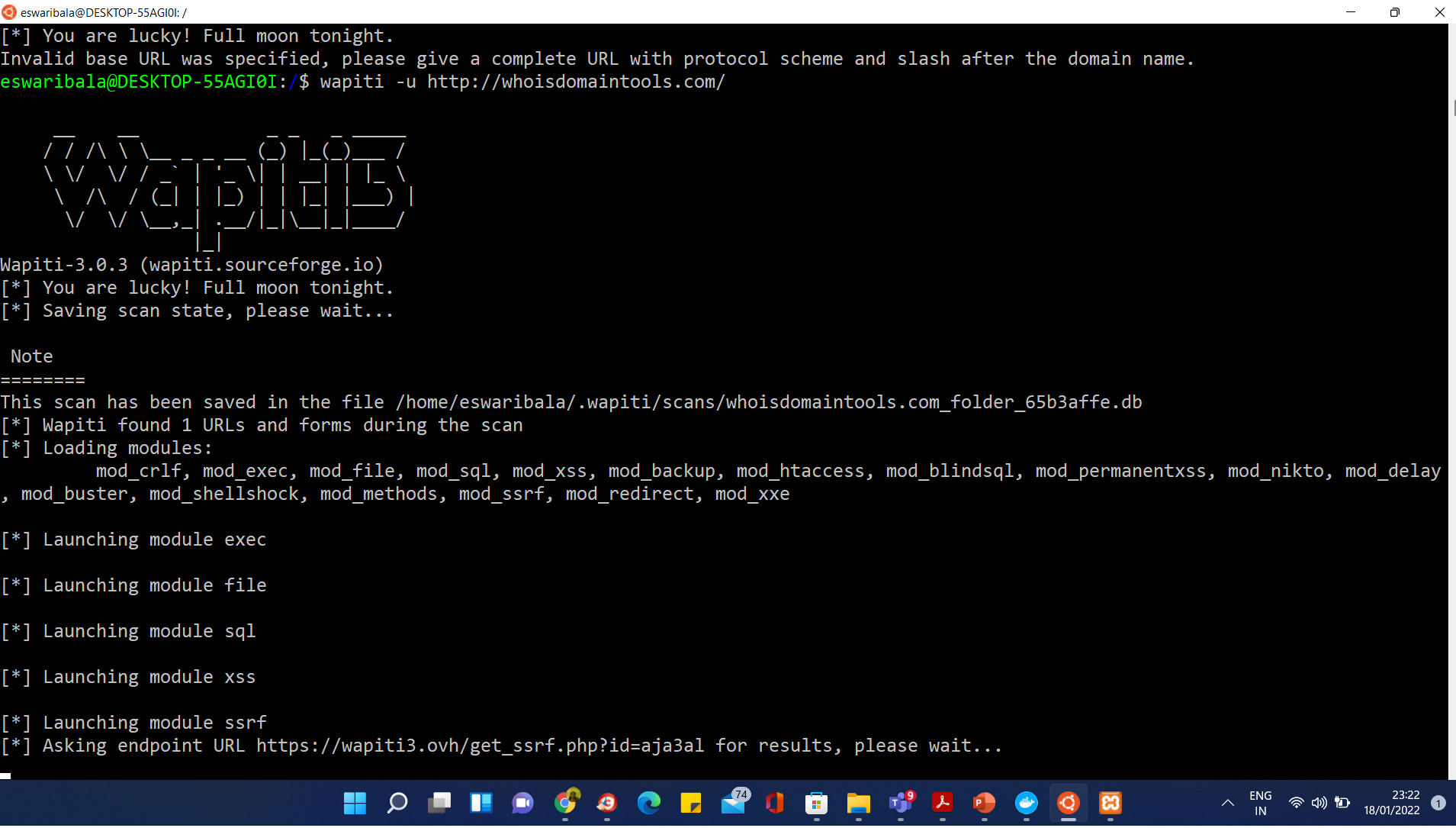
w3af	w3af.org	Open Source	Linux and Mac	GPLv2.0
Wapiti	Informática Gesfor	Open Source	Windows, Unix/Linux and Macintosh	
Web Security Scanner	DefenseCode	Commercial	On-Premises	
WebApp360	TripWire	Commercial	Windows	
WebCookies	WebCookies	Free	SaaS	
WebInspect	Micro Focus	Commercial	Windows	

Vulnerability Tracker Tool

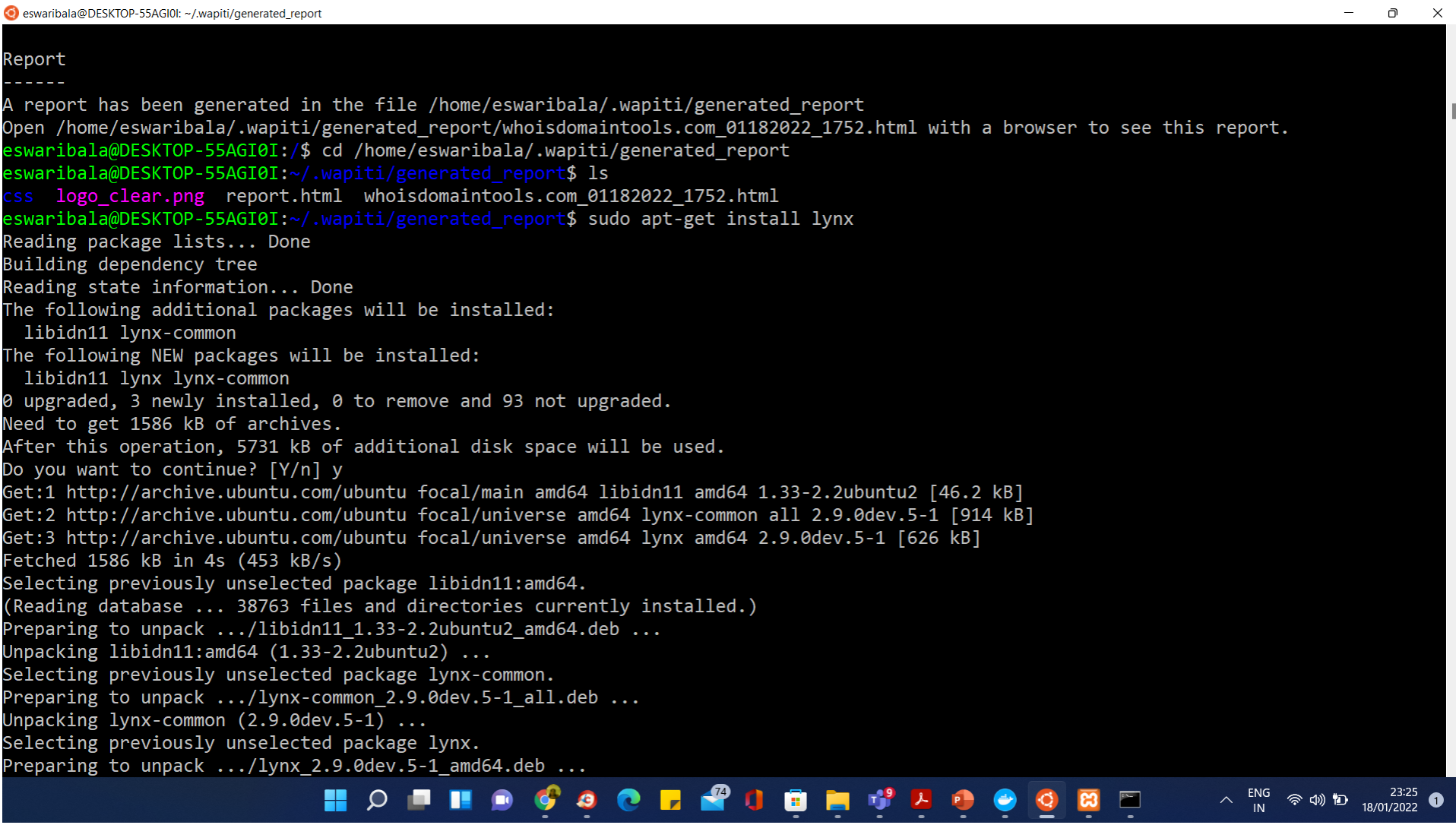
eswaribala@DESKTOP-55AGI0I: /

```
eswaribala@DESKTOP-55AGI0I:/$ sudo apt install wapiti
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  javascript-common libjs-jquery python3-bs4 python3-html5lib python3-lxml python3-mako python3-socks python3-soupsieve python3-tld
  python3-webencodings python3-yaswfp
Suggested packages:
  apache2 | lighttpd | httpd python3-genshi python3-lxml-dbg python-lxml-doc python3-beaker python-mako-doc python-tld-doc
The following NEW packages will be installed:
  javascript-common libjs-jquery python3-bs4 python3-html5lib python3-lxml python3-mako python3-socks python3-soupsieve python3-tld
  python3-webencodings python3-yaswfp wapiti
0 upgraded, 12 newly installed, 0 to remove and 93 not upgraded.
Need to get 2350 kB of archives.
After this operation, 9188 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 javascript-common all 11 [6066 B]
Get:2 http://archive.ubuntu.com/ubuntu focal/main amd64 libjs-jquery all 3.3.1~dfsg-3 [329 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-soupsieve all 1.9.5+dfsg-1 [29.1 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-bs4 all 4.8.2-1 [83.0 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-webencodings all 0.5.1-1ubuntu1 [11.0 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-html5lib all 1.0.1-2 [84.3 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-lxml amd64 4.5.0-1ubuntu0.5 [1384 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-mako all 1.1.0+ds1-1ubuntu2 [59.1 kB]
Get:9 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-socks all 1.6.8+dfsg-1ubuntu1 [18.5 kB]
Get:10 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-tld all 0.9.1-1 [11.5 kB]
Get:11 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-yaswfp all 0.9.3-1 [15.9 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal/universe amd64 wapiti all 3.0.3+dfsg-1 [319 kB]
Fetched 2350 kB in 9s (264 kB/s)
Selecting previously unselected package javascript-common.
(Reading database ... 38336 files and directories currently installed.)
Preparing to unpack .../00-javascript-common_11_all.deb ...
```

Vulnerability Tracker Tool

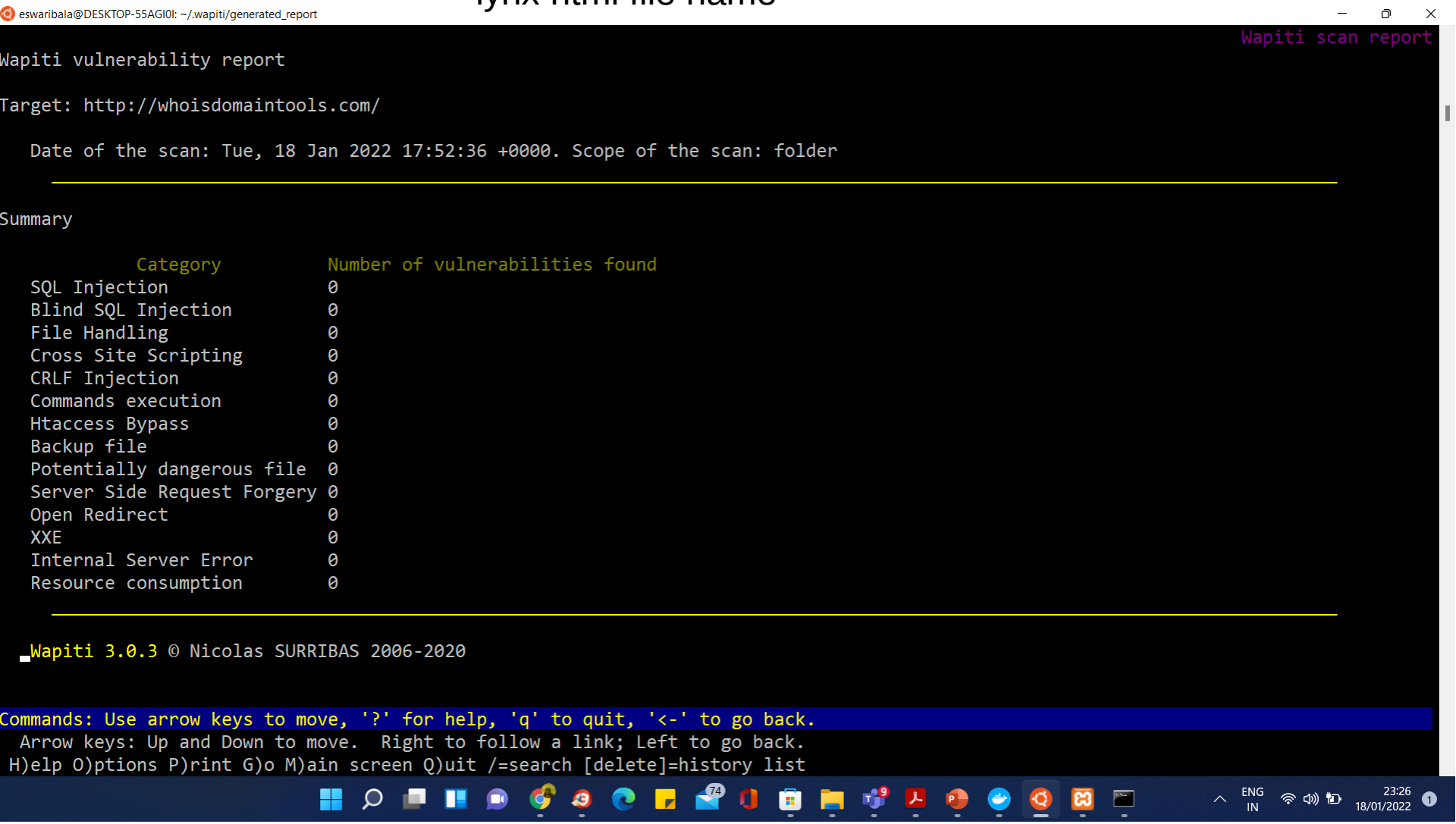


Vulnerability Tracker Tool



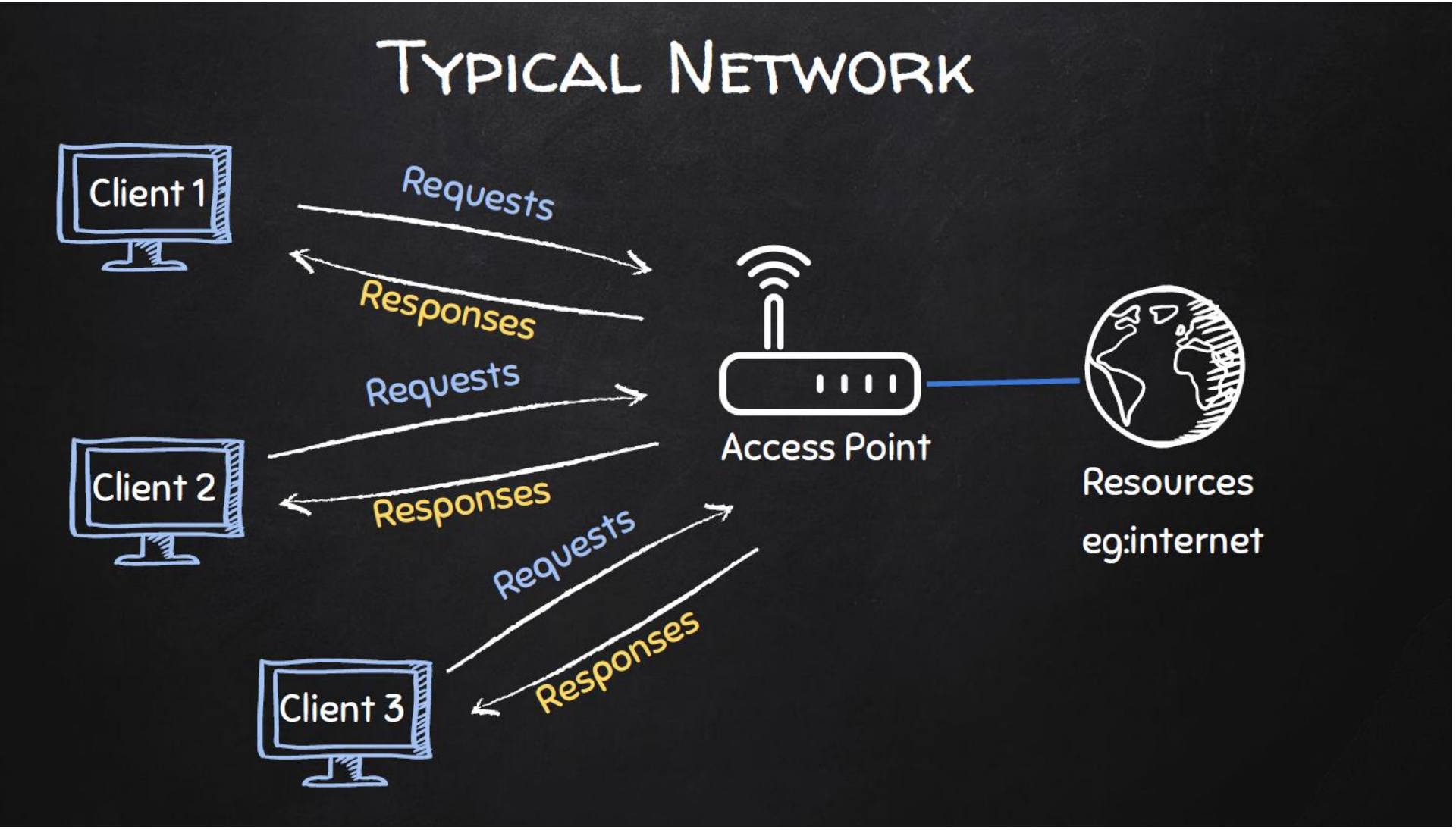
Vulnerability Tracker Tool

lynx html file name

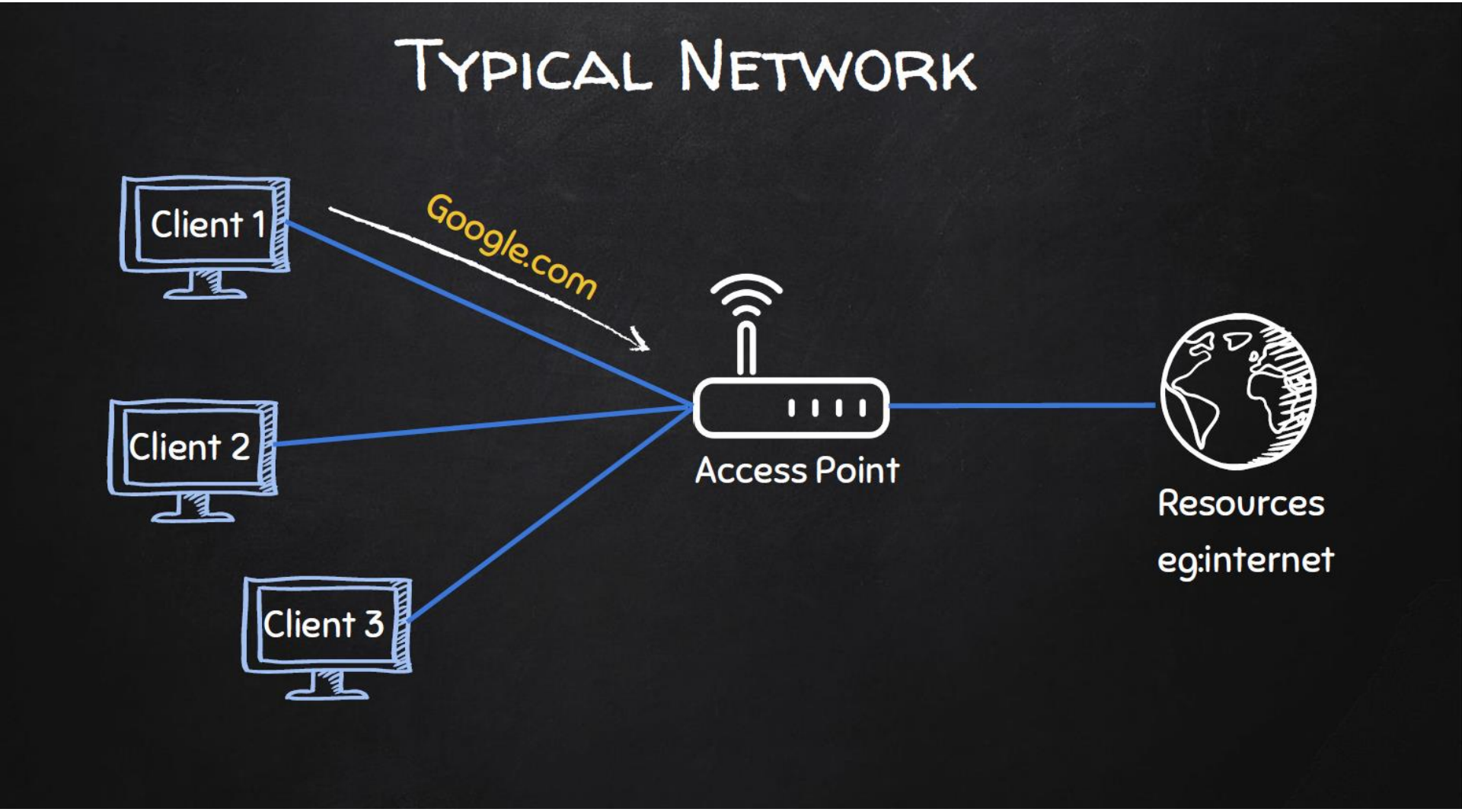


Penetration Testing

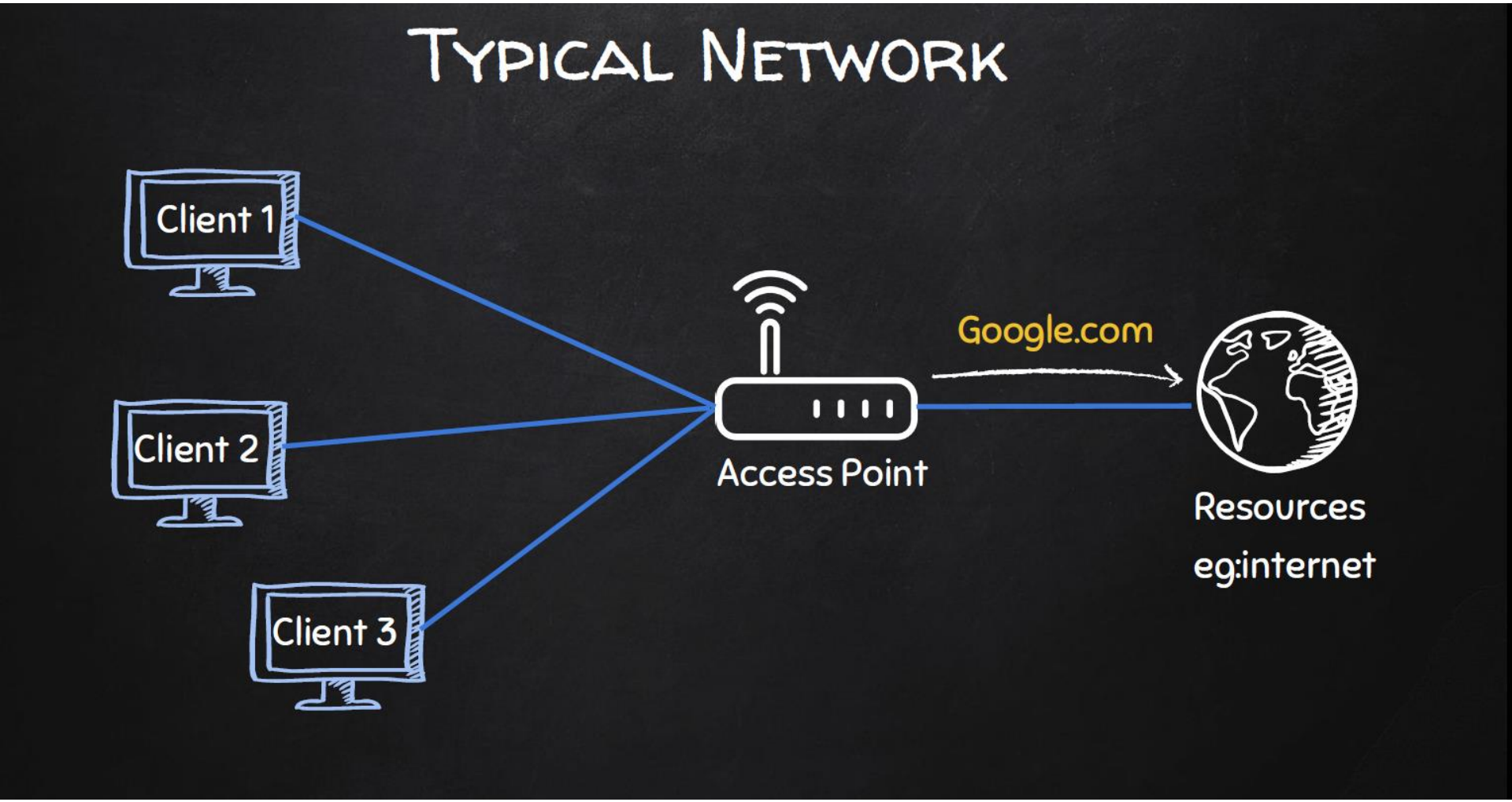
Network Basics



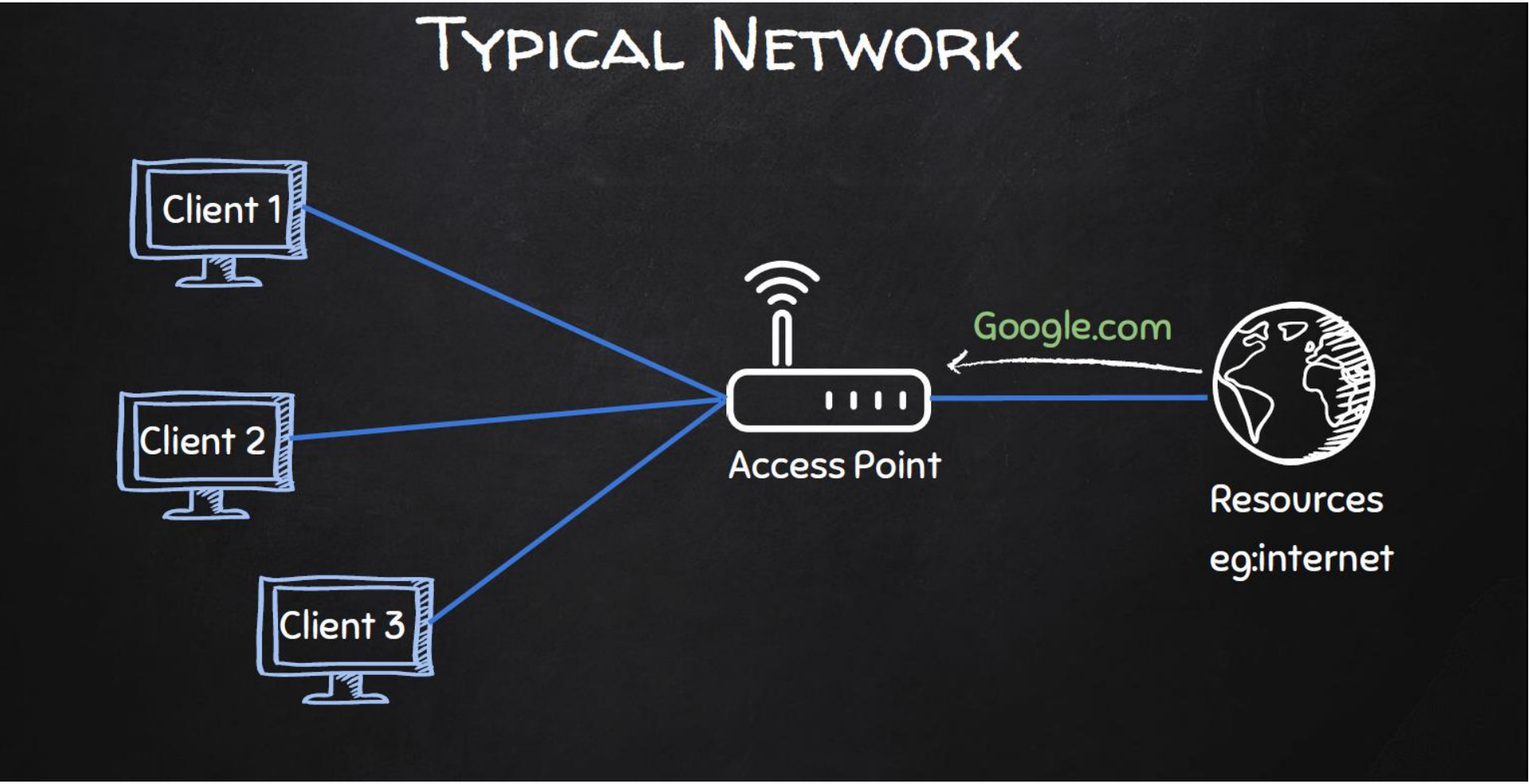
Network Basics



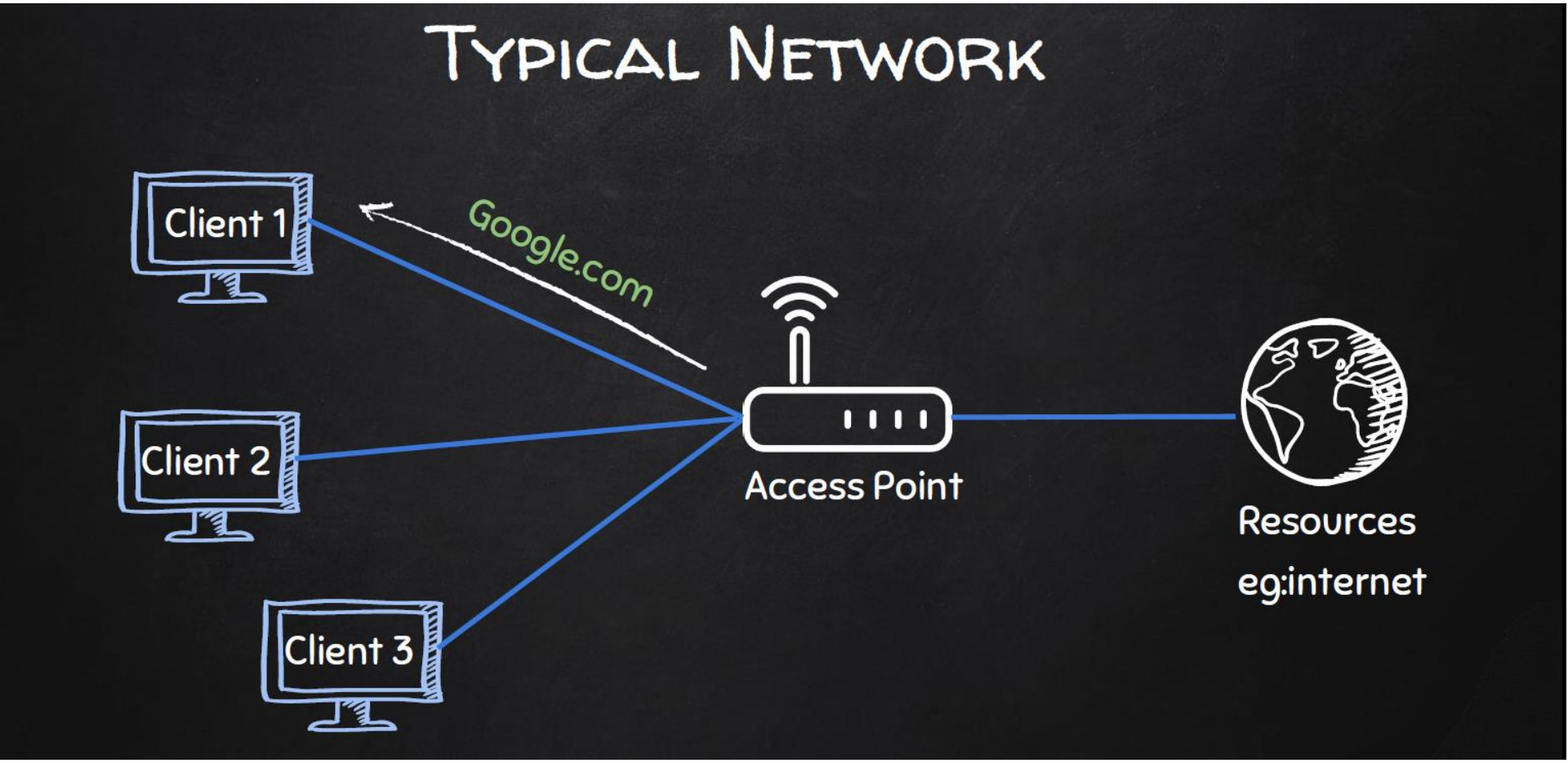
Network Basics



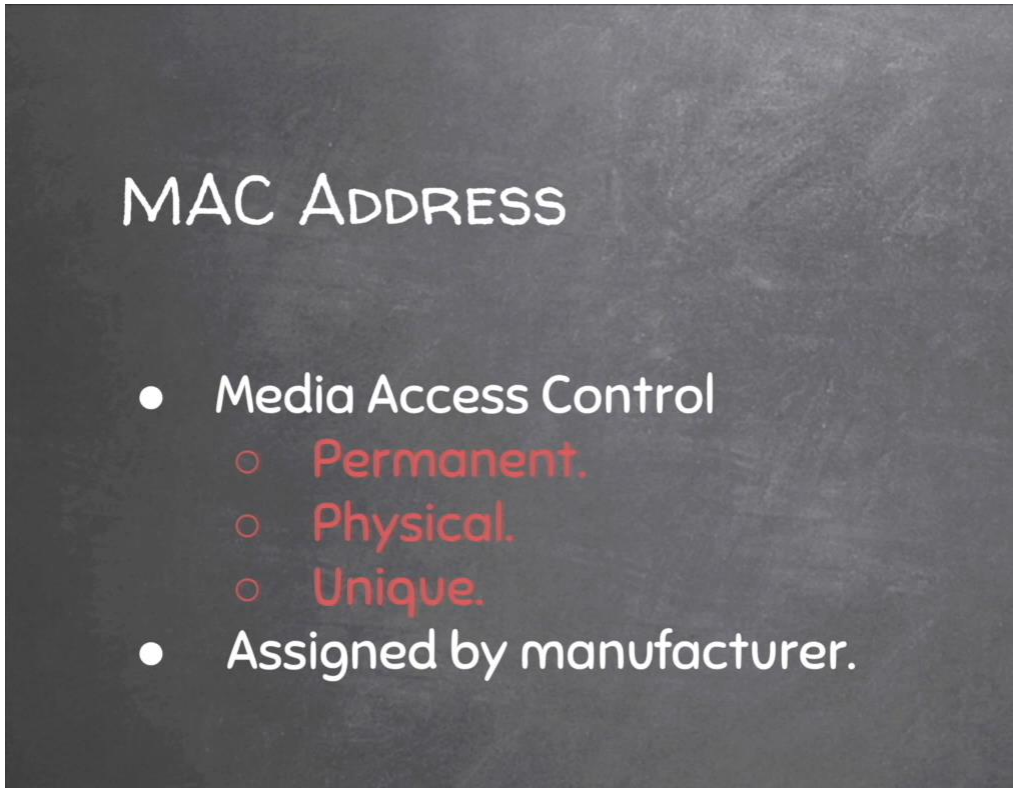
Network Basics



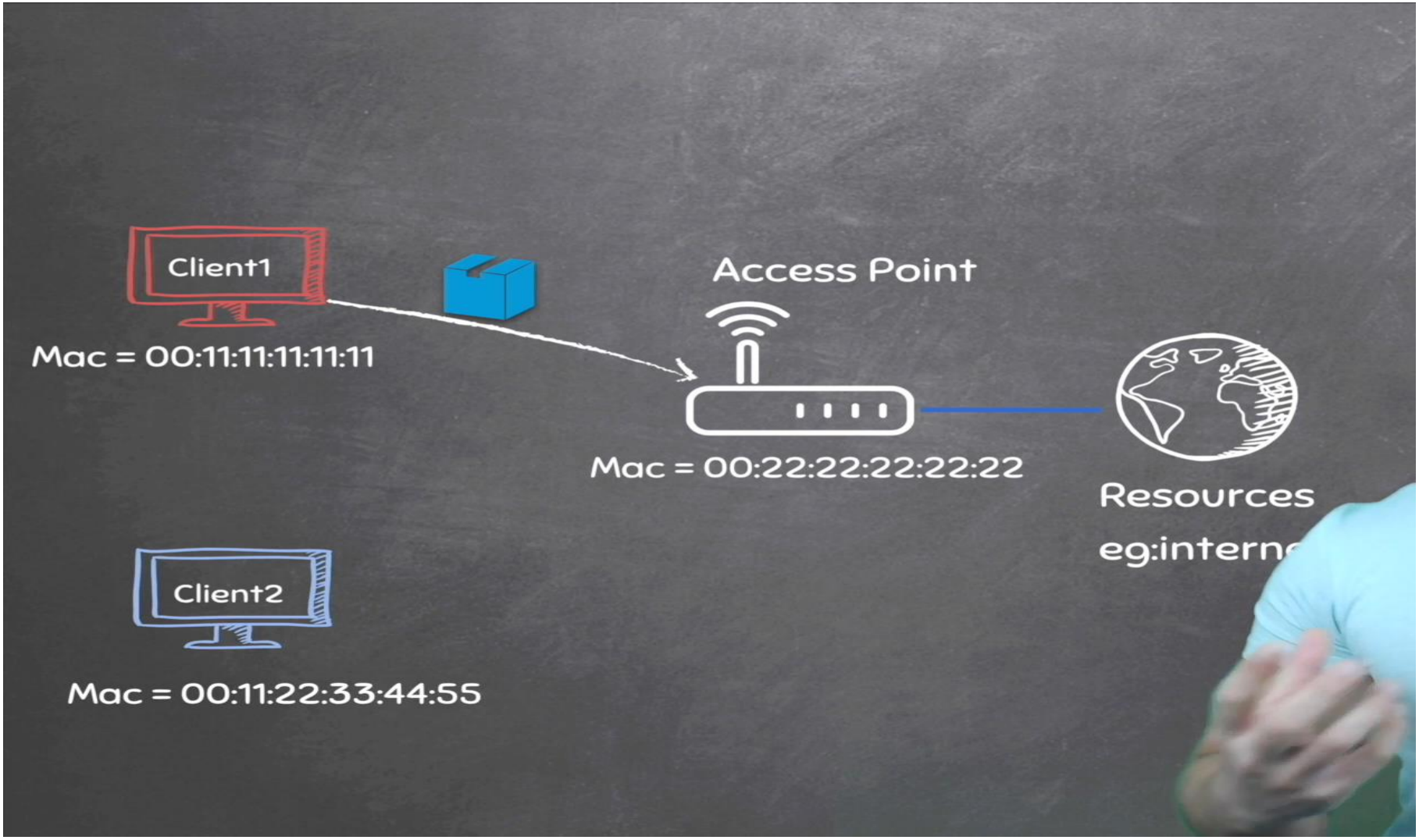
Network Basics



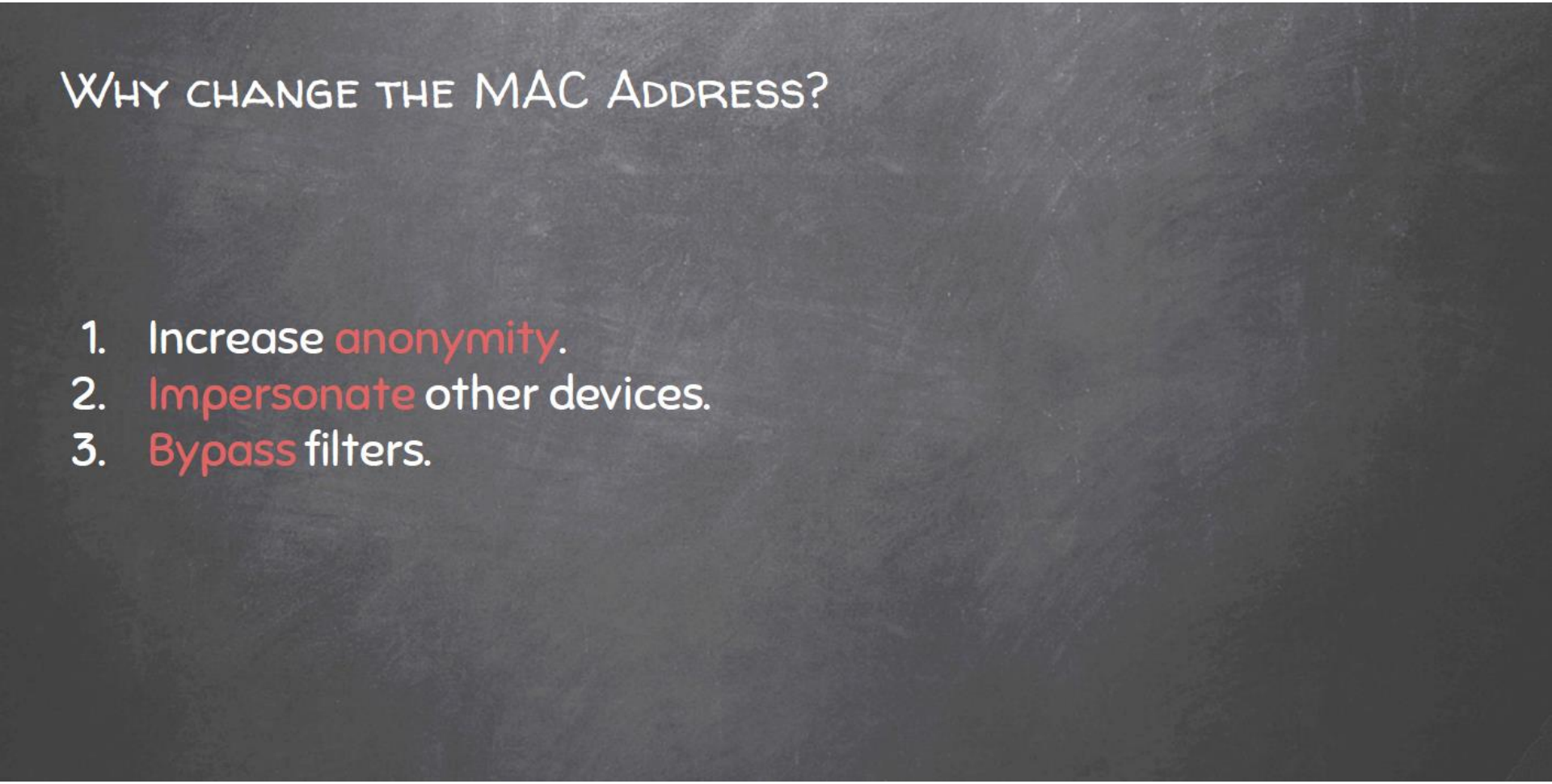
MAC Address


MAC Address
Media Access Control
Permanent.
Physical.
Unique.
Assigned by manufacturer.

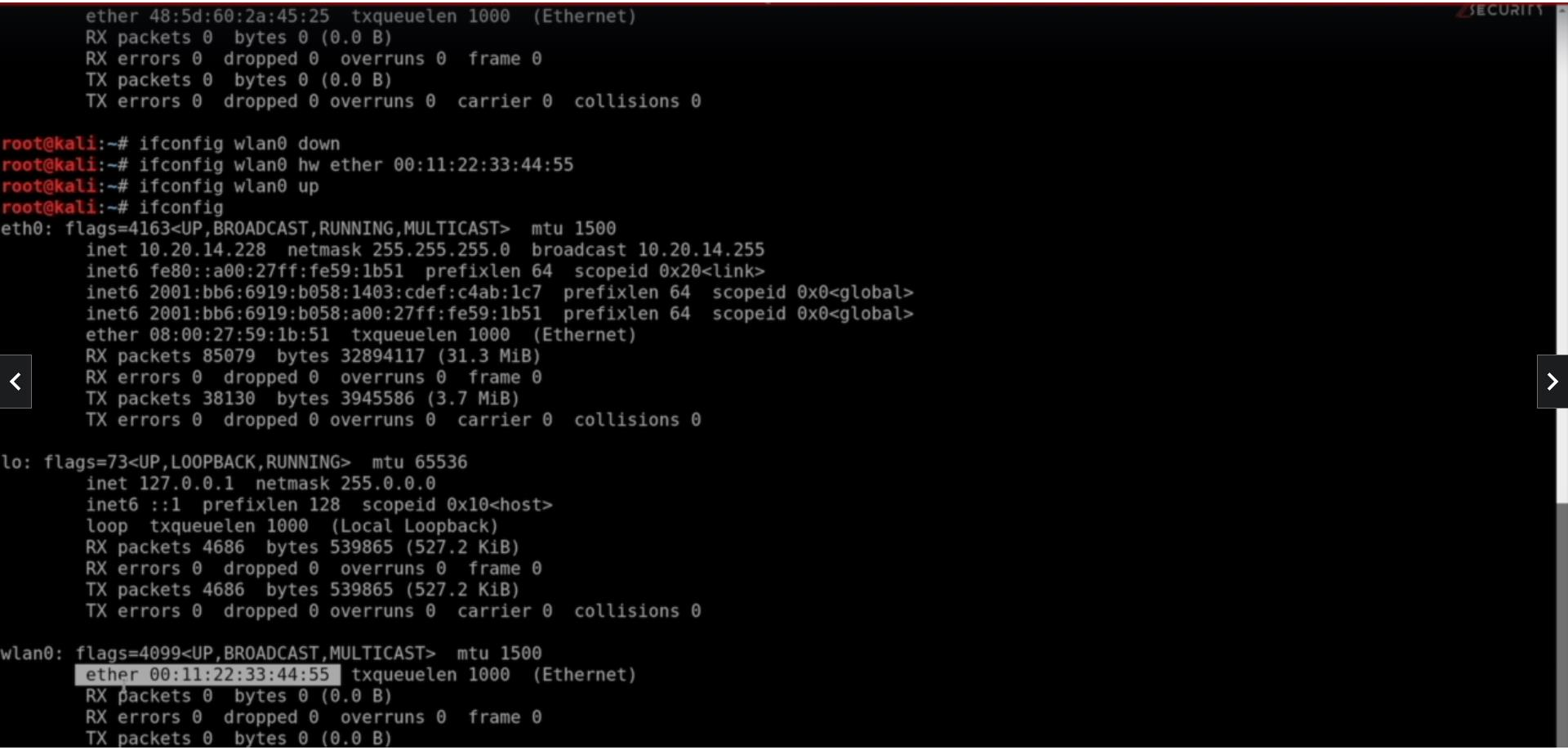
MAC Address


Client1
Mac = 00:11:11:11:11:11
Access Point
Mac = 00:22:22:22:22:22
Resources
Client2
Mac = 00:11:22:33:44:55

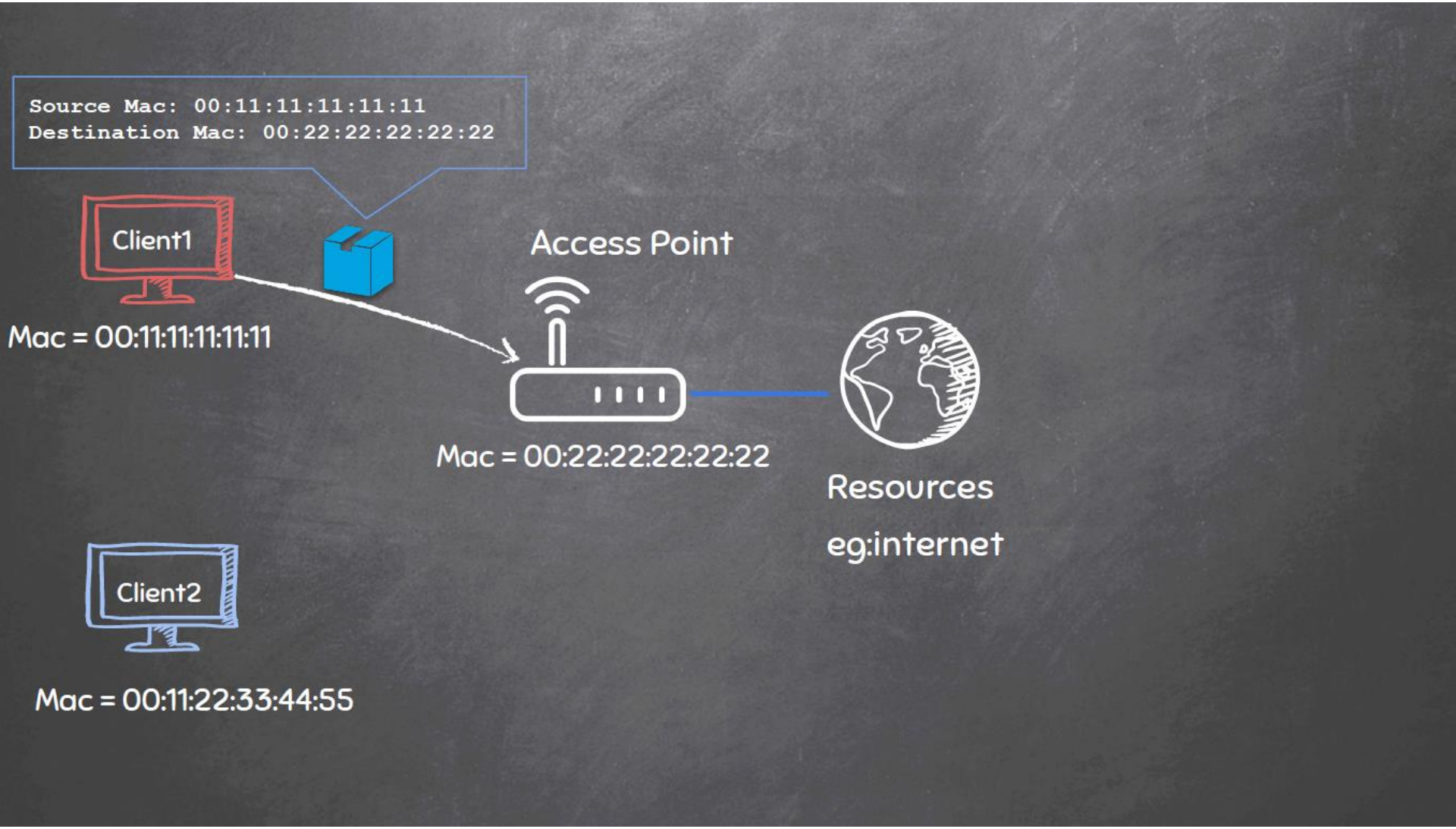
MAC Address


Why change the MAC Address?
1. Increase anonymity.
2. Impersonate other devices.
3. Bypass filters.

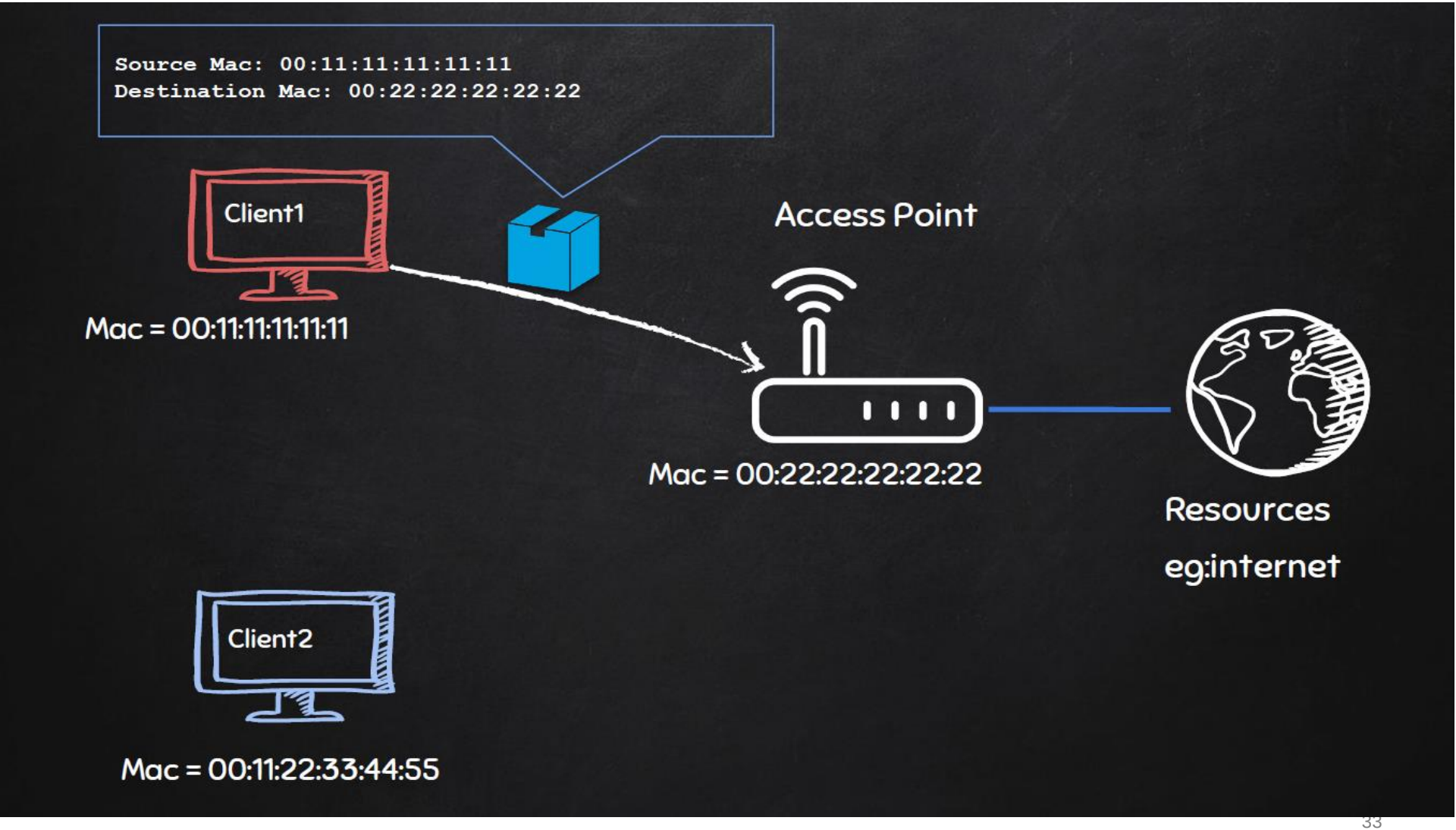
MAC Address


ether 48:5d:60:2a:45:25 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@kali:~# ifconfig wlan0 down
root@kali:~# ifconfig wlan0 hw ether 00:11:22:33:44:55
root@kali:~# ifconfig wlan0 up
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.20.14.228 netmask 255.255.255.0 broadcast 10.20.14.255
inet6 fe80::a00:27ff:fe59:1b51 prefixlen 64 scopeid 0x20<link>
inet6 2001:bb6:6919:b058:1403:cdef:c4ab:1c7 prefixlen 64 scopeid 0x0<global>
inet6 2001:bb6:6919:b058:a00:27ff:fe59:1b51 prefixlen 64 scopeid 0x0<global>
ether 08:00:27:59:1b:51 txqueuelen 1000 (Ethernet)
RX packets 85079 bytes 32894117 (31.3 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 38130 bytes 3945586 (3.7 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 4686 bytes 539865 (527.2 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 4686 bytes 539865 (527.2 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
wlan0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
ether 00:11:22:33:44:55 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)

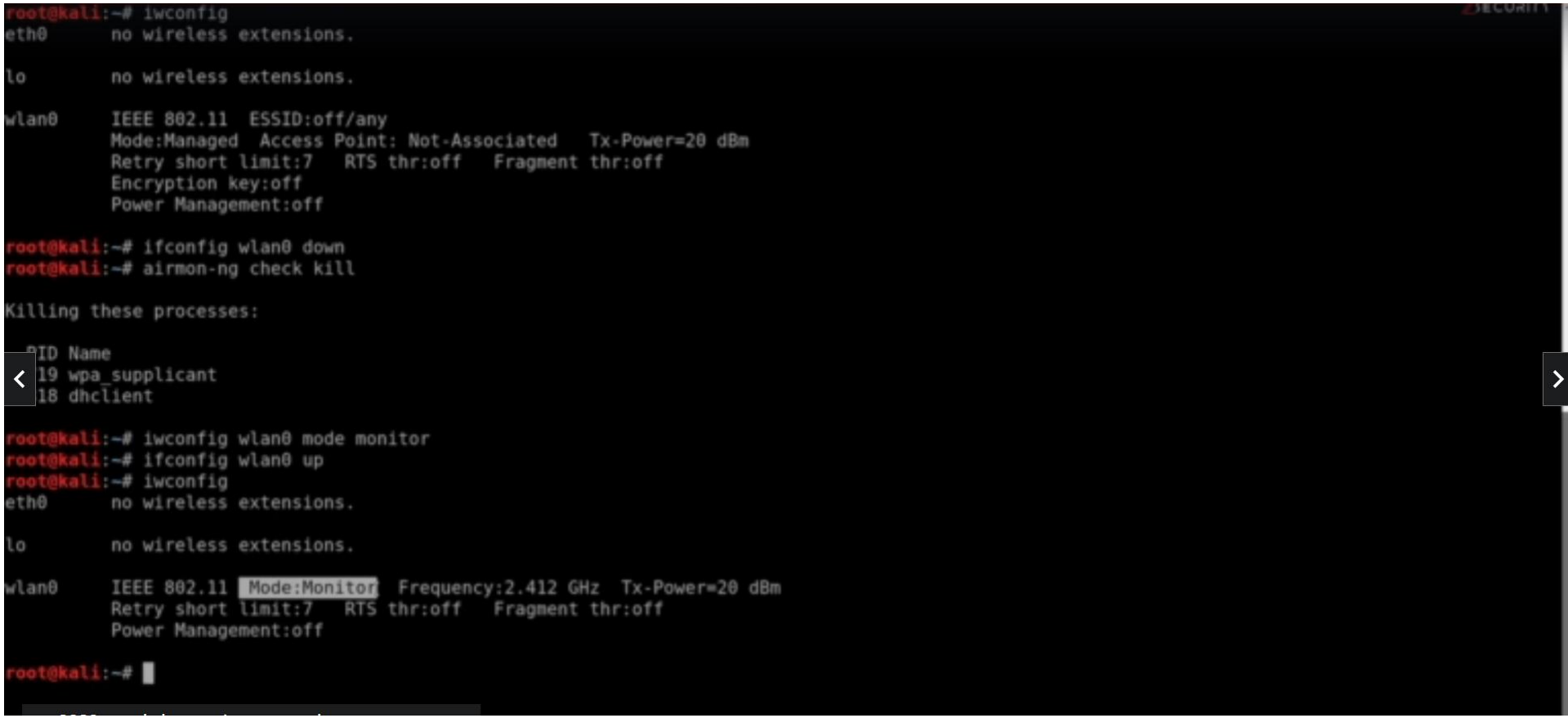
MAC Address


Source Mac: 00:11:11:11:11:11
Destination Mac: 00:22:22:22:22:22
Client1
Mac = 00:11:11:11:11:11
Access Point
Mac = 00:22:22:22:22:22
Resources
eg:internet
Client2
Mac = 00:11:22:33:44:55

MAC Address


Source Mac: 00:11:11:11:11:11
Destination Mac: 00:22:22:22:22:22
Client1
Mac = 00:11:11:11:11:11
Access Point
Mac = 00:22:22:22:22:22
Resources
eg:internet
Client2
Mac = 00:11:22:33:44:55

MAC Address


root@kali:~# iwconfig
eth0 no wireless extensions.
lo no wireless extensions.
wlan0 IEEE 802.11 ESSID:off/any
Mode:Managed Access Point: Not-Associated Tx-Power=20 dBm
Retry short limit:7 RTS thr:off Fragment thr:off
Encryption key:off
Power Management:off
root@kali:~# ifconfig wlan0 down
root@kali:~# airmon-ng check kill
Killing these processes:
PID Name
719 wpa_supplicant
718 dhclient
root@kali:~# iwconfig wlan0 mode monitor
root@kali:~# ifconfig wlan0 up
root@kali:~# iwconfig
eth0 no wireless extensions.
lo no wireless extensions.
wlan0 IEEE 802.11 Mode:Monitor Frequency:2.412 GHz Tx-Power=20 dBm
Retry short limit:7 RTS thr:off Fragment thr:off
Power Management:off
root@kali:~#

Packet Sniffing

Packet Sniffing

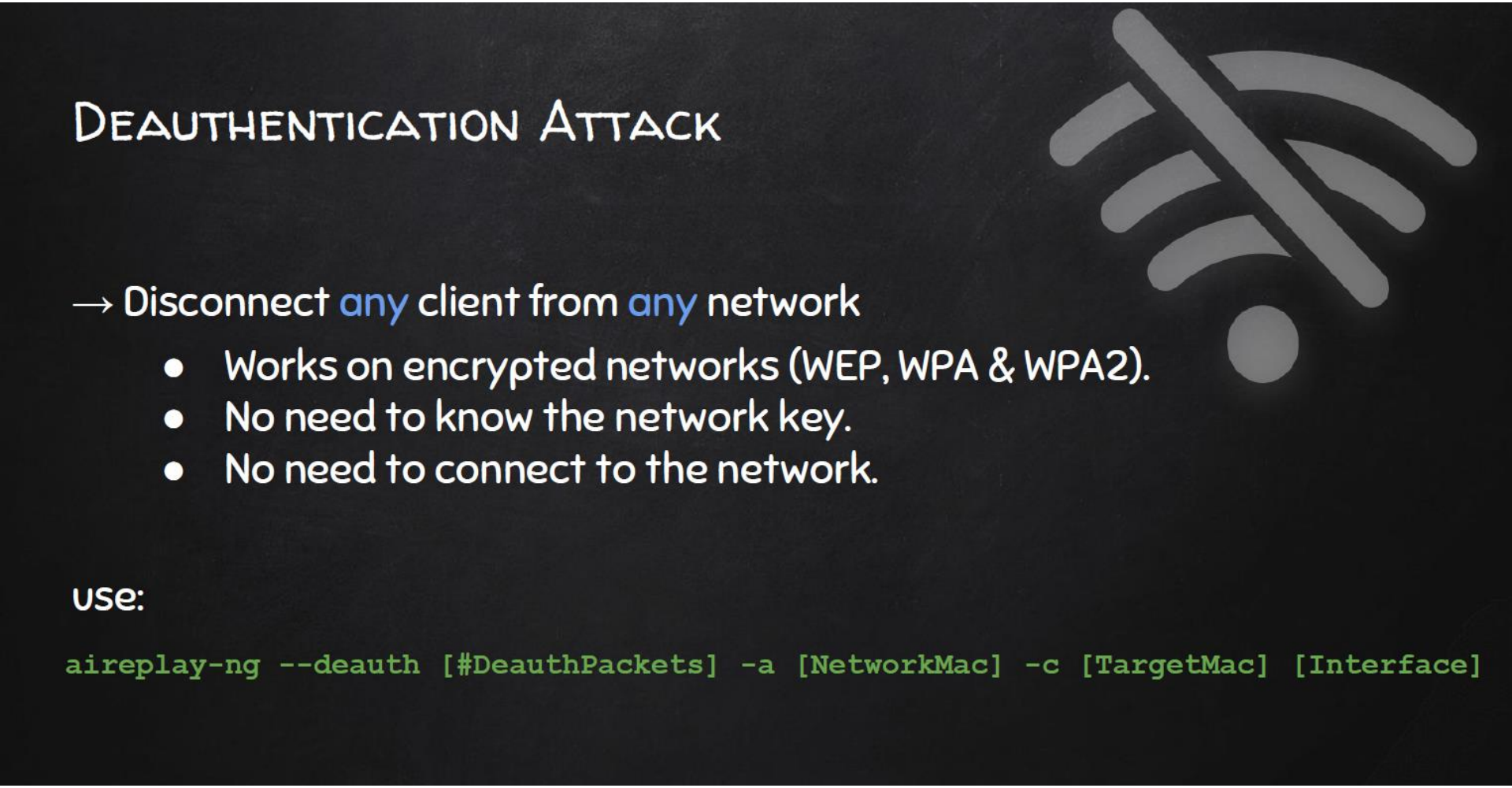
Using Airodump-ng

- Part of the aircrack-ng suit.
- Airodump-ng is a packet sniffer;
- Used to capture all packets within range.
- Display detailed info about networks around us.
- Connected clients ....etc

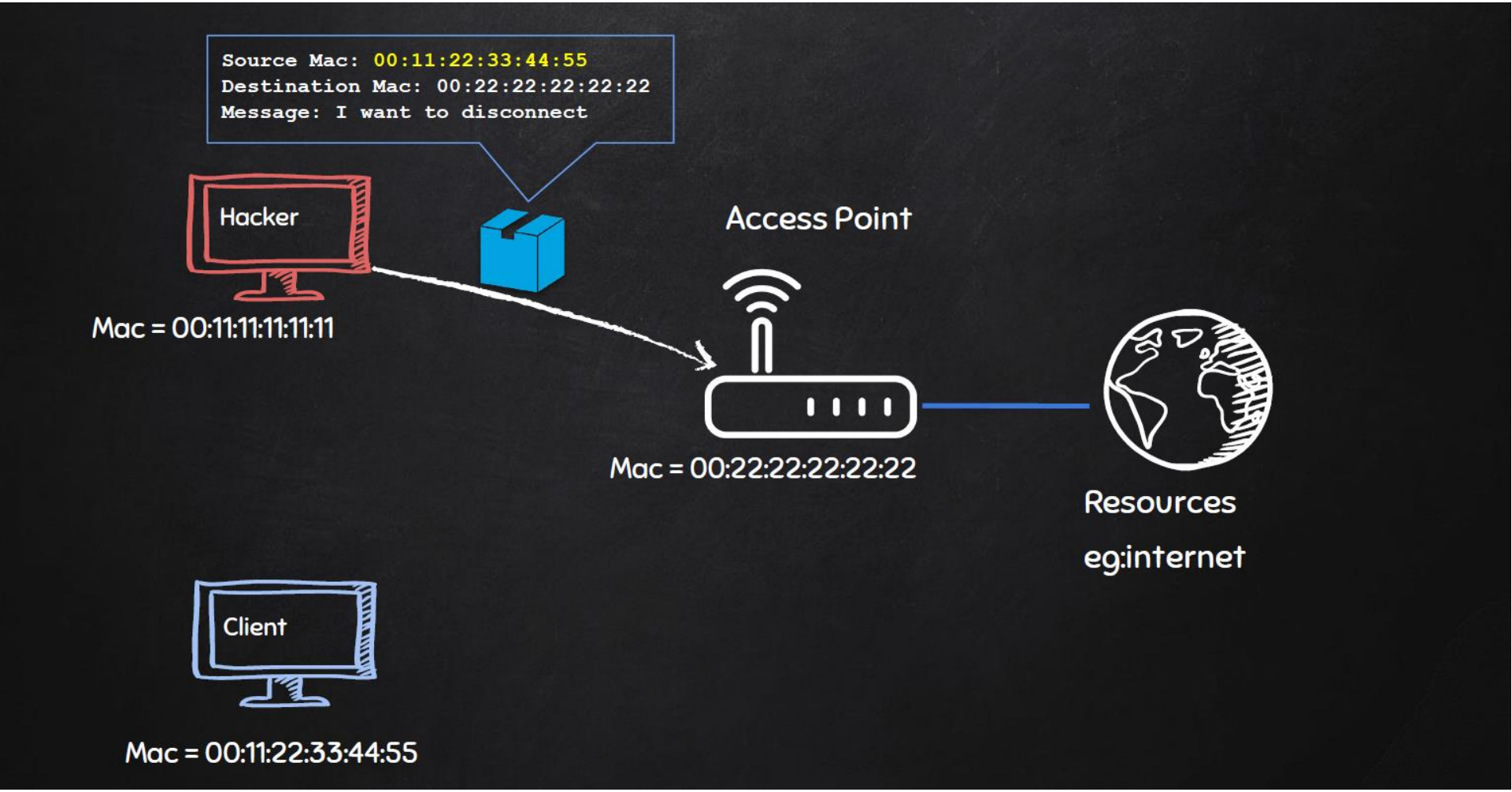
use:

airodump-ng [MonitorModeInterface]

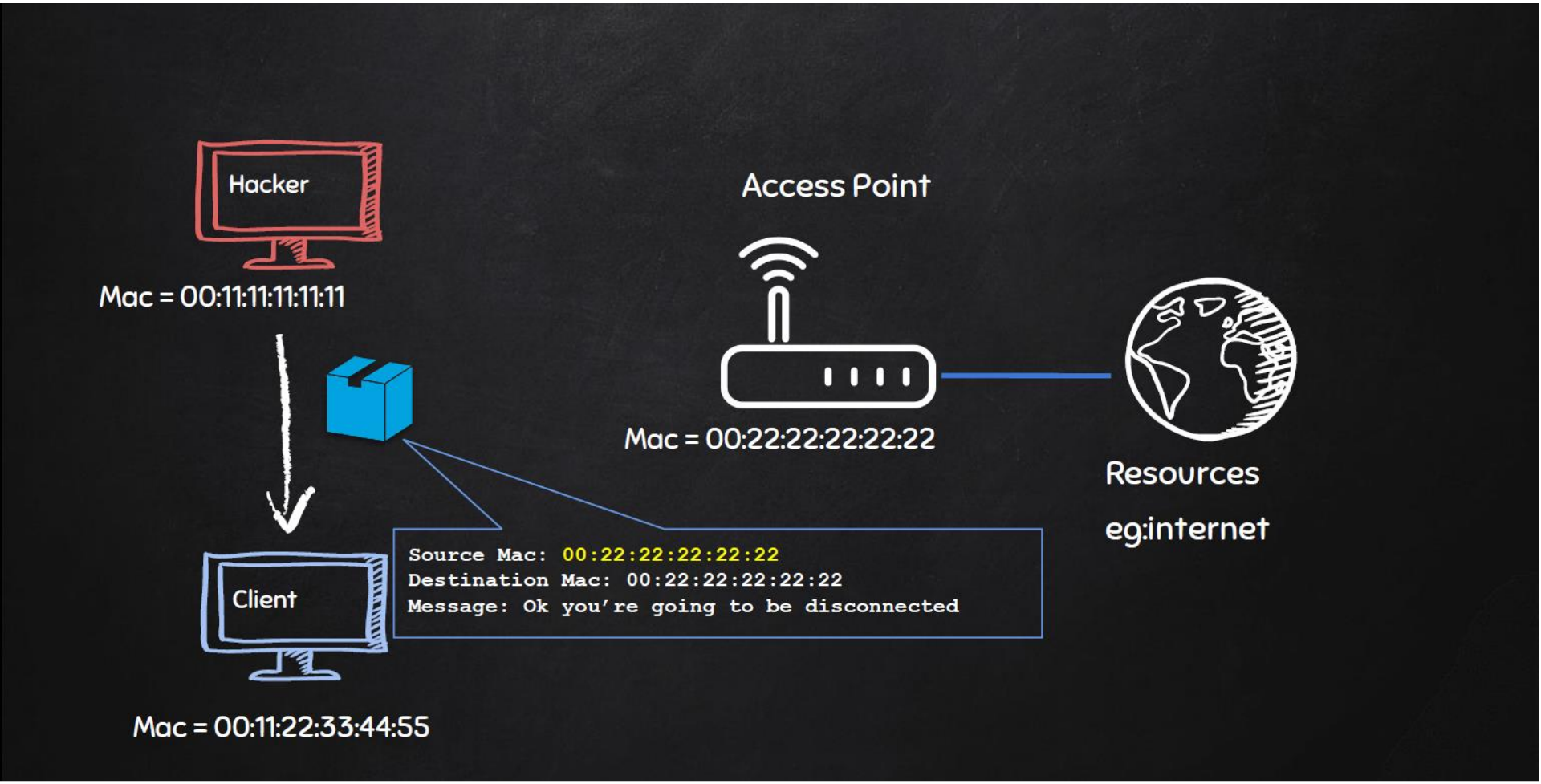
Packet Sniffing



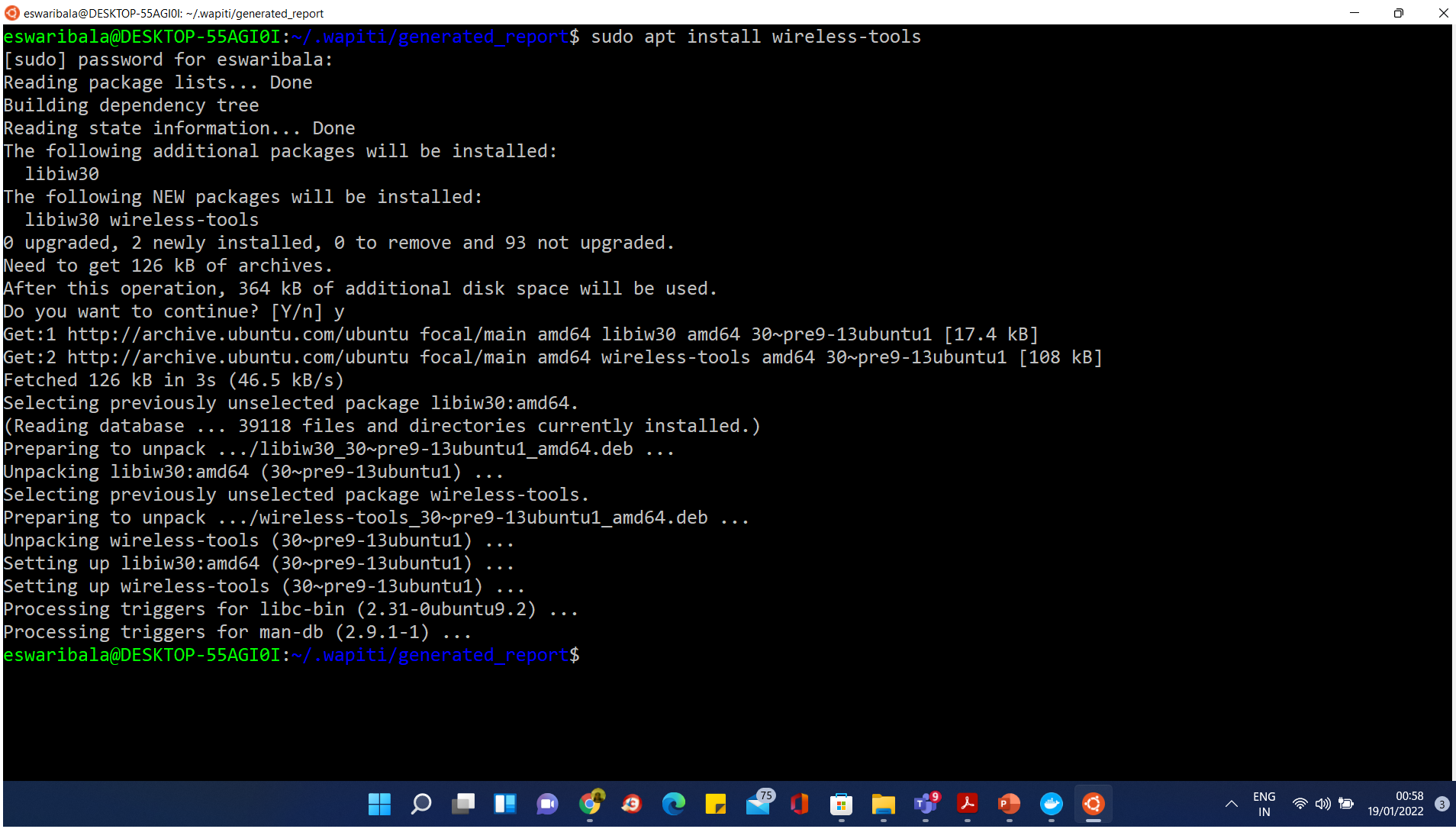
Packet Sniffing



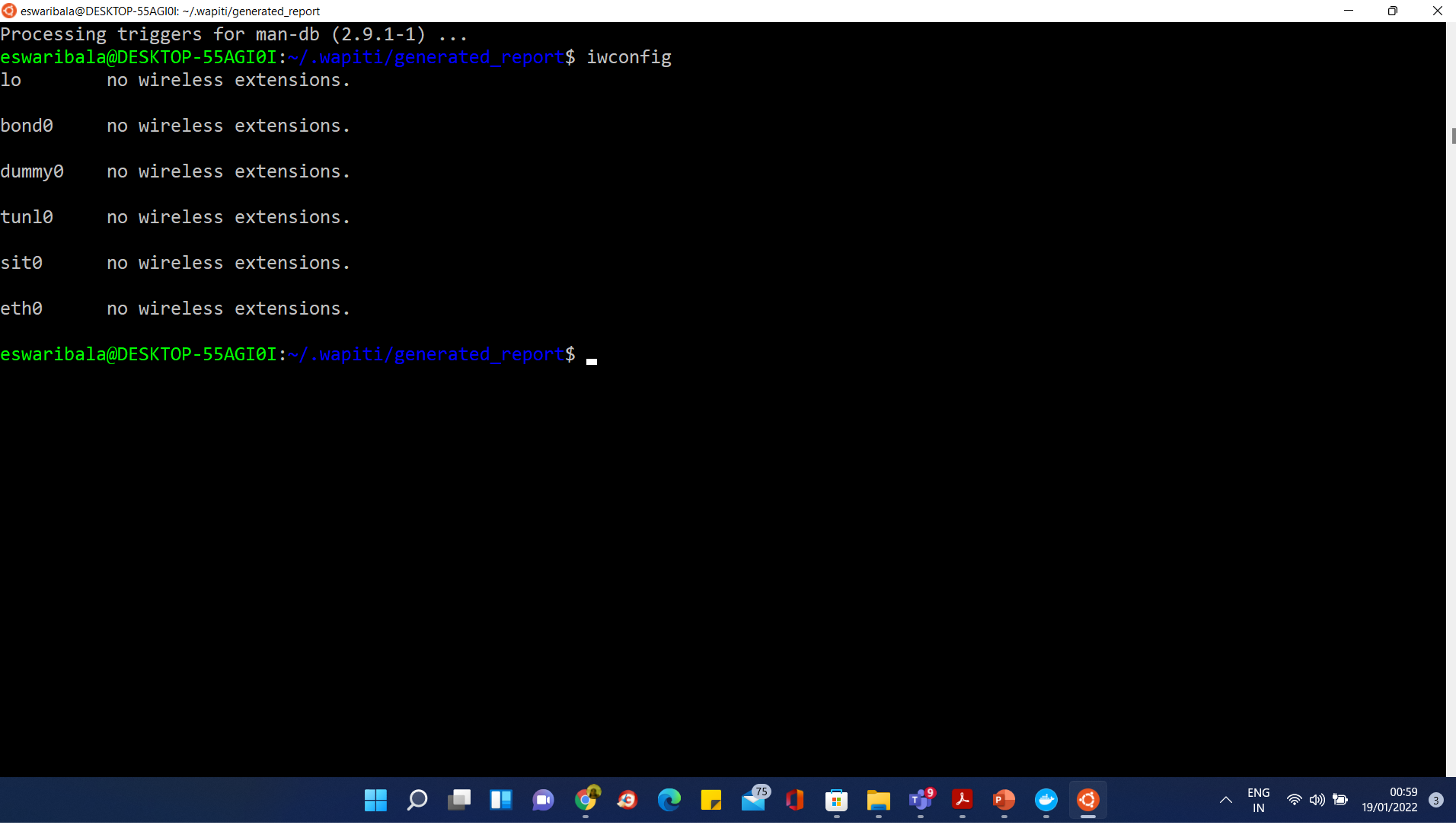
Packet Sniffing



Packet Sniffing Basics



Packet Sniffing Basics



Packet Sniffing Basics

eswaribala@DESKTOP-55AGI0I: ~/.wapiti/generated_report

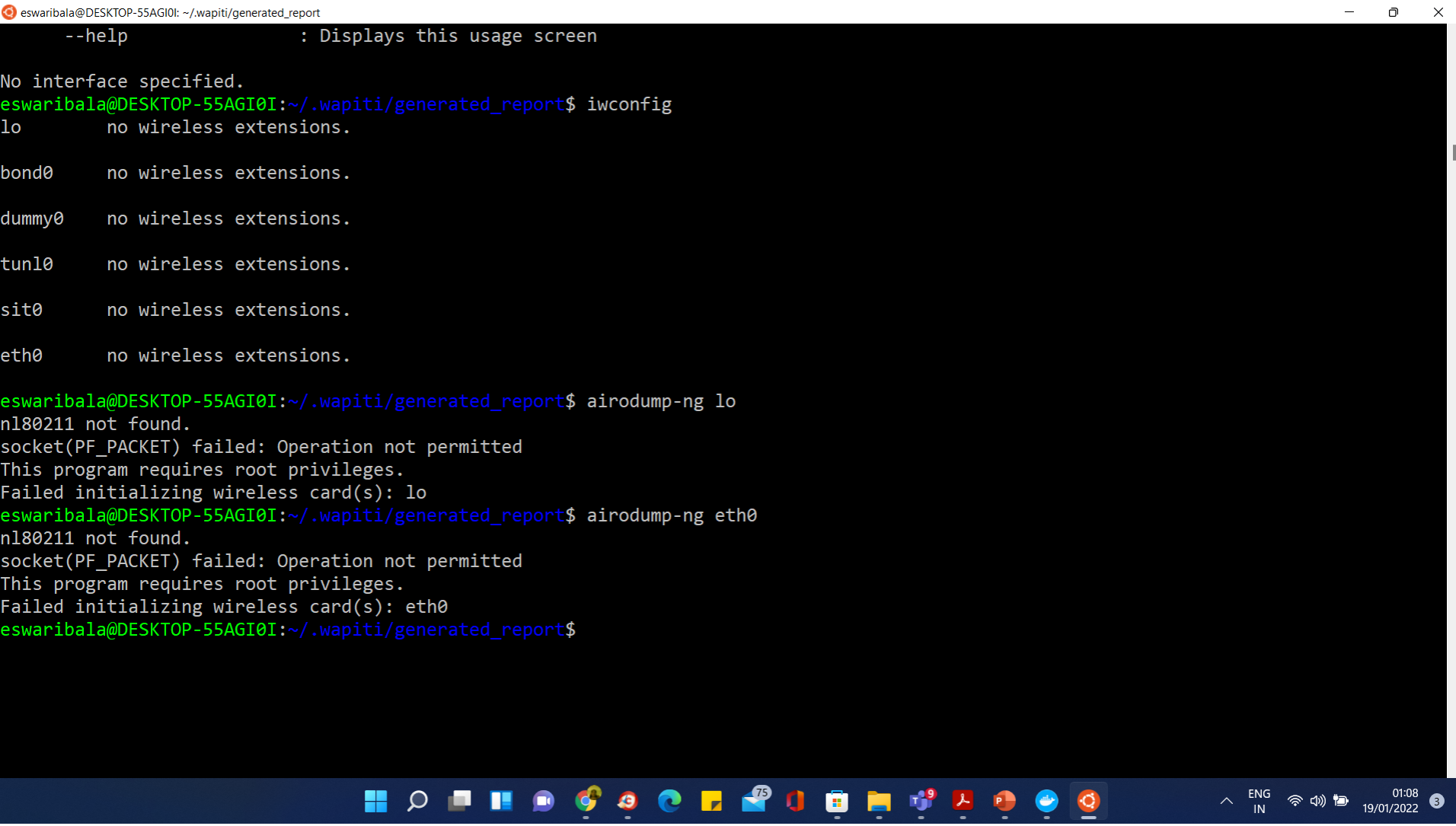
```
eswaribala@DESKTOP-55AGI0I:~/.wapiti/generated_report$ sudo apt install aircrack-ng
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  crda hwloc ieee-data iw libcairo2 libhwloc-plugins libhwloc15 libnl-3-200 libnl-genl-3-200 libpixman-1-0 libxcb-render0 libxnvctrl0
  ocl-icd-libopencl1 rfkill wireless-regdb
Suggested packages:
  gpsd libhwloc-contrib-plugins opencl-icd
The following NEW packages will be installed:
  aircrack-ng crda hwloc ieee-data iw libcairo2 libhwloc-plugins libhwloc15 libnl-3-200 libnl-genl-3-200 libpixman-1-0 libxcb-render0
  libxnvctrl0 ocl-icd-libopencl1 rfkill wireless-regdb
0 upgraded, 16 newly installed, 0 to remove and 93 not upgraded.
Need to get 3541 kB of archives.
After this operation, 17.3 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 libpixman-1-0 amd64 0.38.4-0ubuntu1 [227 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal/main amd64 libxcb-render0 amd64 1.14-2 [14.8 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal/main amd64 libcairo2 amd64 1.16.0-4ubuntu1 [583 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal/universe amd64 libhwloc15 amd64 2.1.0+dfsg-4 [134 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal/universe amd64 hwloc amd64 2.1.0+dfsg-4 [174 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/main amd64 libnl-3-200 amd64 3.4.0-1 [53.9 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/main amd64 libnl-genl-3-200 amd64 3.4.0-1 [11.1 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal/main amd64 iw amd64 5.4-1 [94.0 kB]
Get:9 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 rfkill amd64 2.34-0.1ubuntu9.1 [22.9 kB]
Get:10 http://archive.ubuntu.com/ubuntu focal/universe amd64 aircrack-ng amd64 1:1.6-4 [508 kB]
Get:11 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 wireless-regdb all 2021.08.28-0ubuntu1~20.04.1 [10.0 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal/main amd64 crda amd64 3.18-1build1 [63.5 kB]
Get:13 http://archive.ubuntu.com/ubuntu focal/main amd64 ieee-data all 20180805.1 [1589 kB]
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libxnvctrl0 amd64 470.57.01-0ubuntu0.20.04.2 [11.1 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal/main amd64 ocl-icd-libopencl1 amd64 2.2.11-1ubuntu1 [30.3 kB]
Get:16 http://archive.ubuntu.com/ubuntu focal/universe amd64 libhwloc-plugins amd64 2.1.0+dfsg-4 [14.4 kB]
```

Packet Sniffing Basics

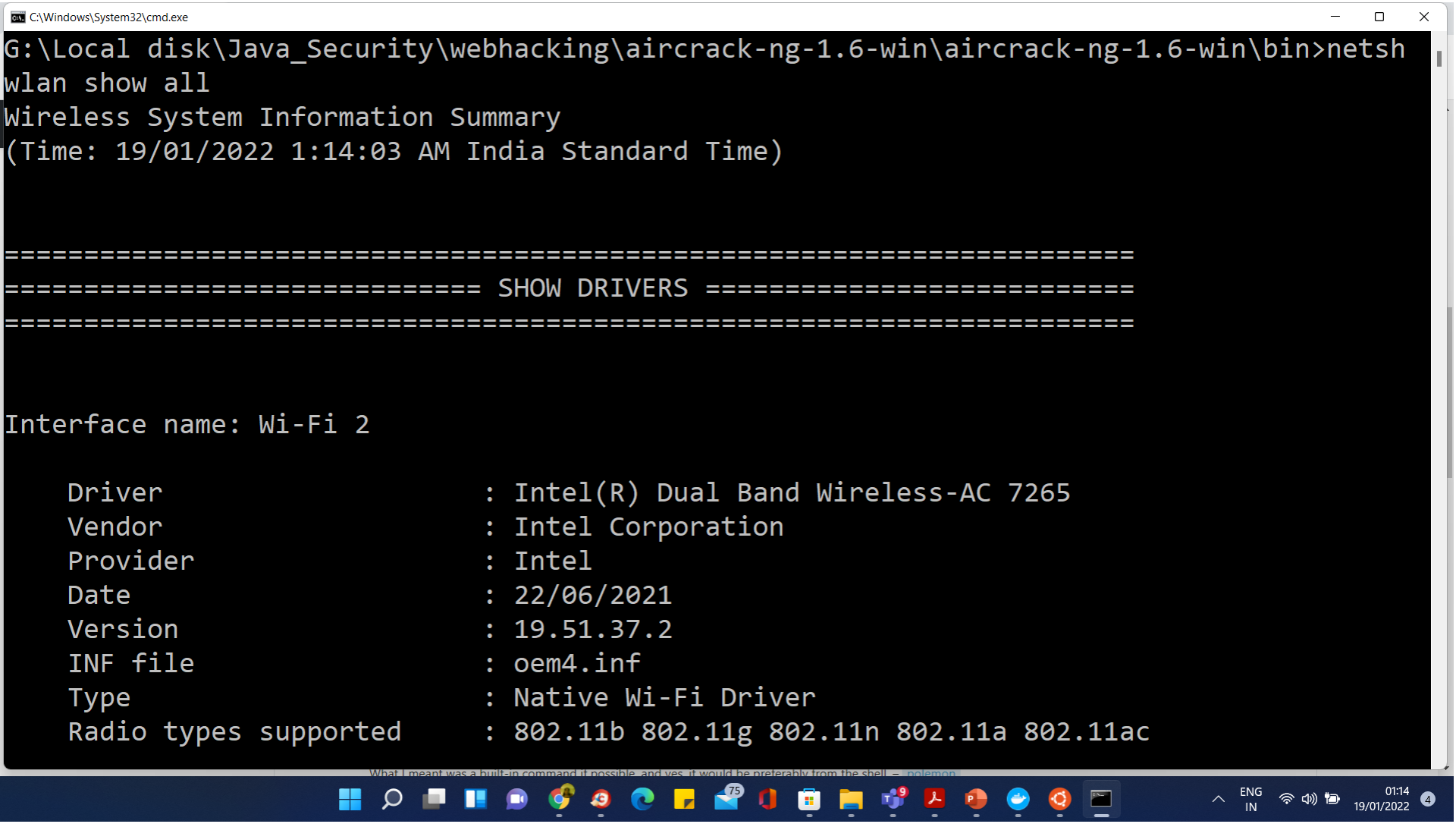
eswaribala@DESKTOP-55AGI0I: ~/.wapiti/generated_report

```
eswaribala@DESKTOP-55AGI0I:~/.wapiti/generated_report$ airodump-ng

 Airodump-ng 1.6  - (C) 2006-2020 Thomas d'Otreppe
 https://www.aircrack-ng.org

 usage: airodump-ng <options> <interface>[,<interface>,...]

 Options:
     --ivs                : Save only captured IVs
     --gpsd               : Use GPSd
     --write      <prefix> : Dump file prefix
     -w                    : same as --write
     --beacons            : Record all beacons in dump file
     --update       <secs> : Display update delay in seconds
     --showack            : Prints ack/cts/rts statistics
     -h                   : Hides known stations for --showack
     -f            <msecs> : Time in ms between hopping channels
     --berlin       <secs> : Time before removing the AP/client
                             from the screen when no more packets
                             are received (Default: 120 seconds)
     -r             <file> : Read packets from that file
     -T                    : While reading packets from a file,
                             simulate the arrival rate of them
                             as if they were "live".
     -x            <msecs> : Active Scanning Simulation
     --manufacturer       : Display manufacturer from IEEE OUI list
     --uptime             : Display AP Uptime from Beacon Timestamp
     --wps                : Display WPS information (if any)
     --output-format
                 <formats> : Output format. Possible values:
                             pcap, ivs, csv, gps, kismet, netxml, logcsv
     --ignore-negative-one : Removes the message that says
                             fixed channel <interface>: -1
```

Packet Sniffing Basics

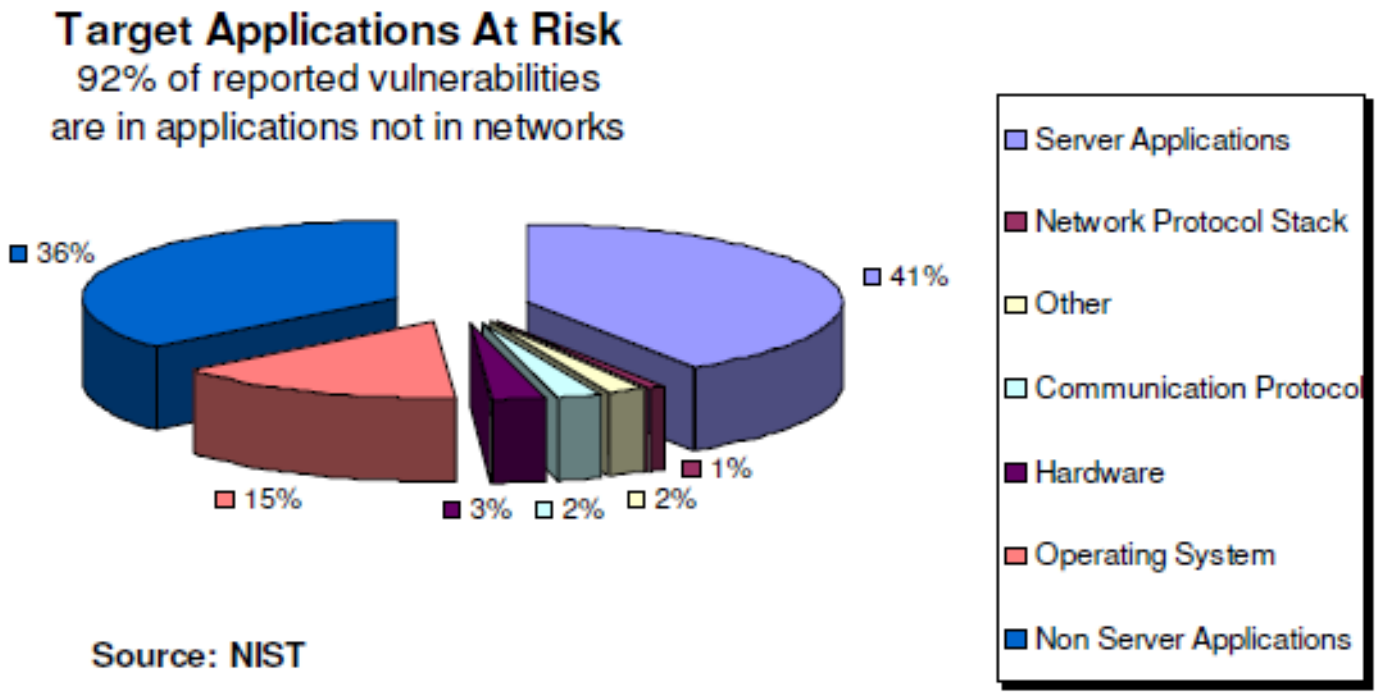


Packet Sniffing Basics



Applications under Attack

What is at risk?



Source: NIST

Applications under Attack

How we approach risk?

Application Security

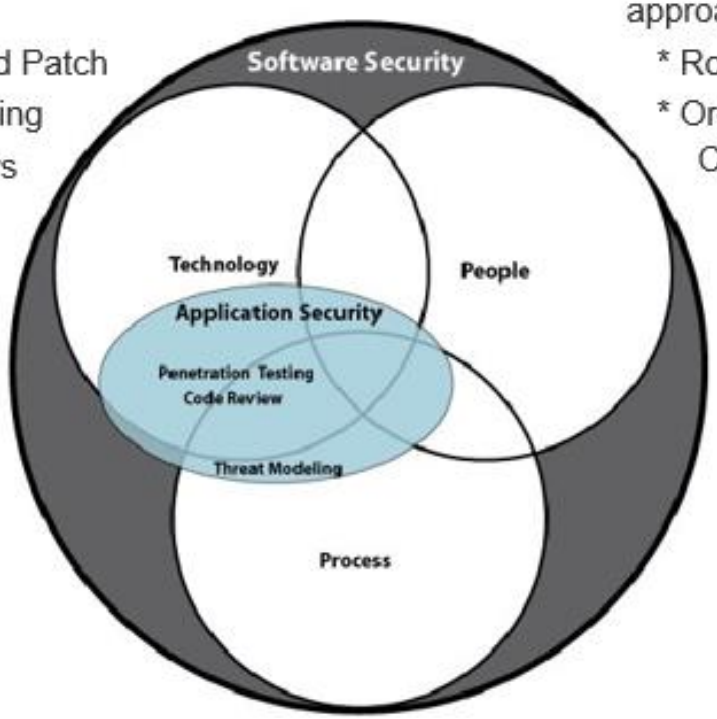
– Issue-based, short-term approach

* Penetrate and Patch
* Threat Modeling
* Code Reviews

Software Security

– Holistic, long-term approach

* Root Cause Analysis
* Organizational Change



Applications under Attack

Application Security Costs:

» Defect Management: 5 defects/KLOC, $ 30,000/KLOC (Business week)

» Patch Management: 1000 servers, $ 300,000 to test and deploy a patch (Gartner)

» Loss of productivity due of loss of service: $ 500 ML lost from DoS attack (Microsoft)

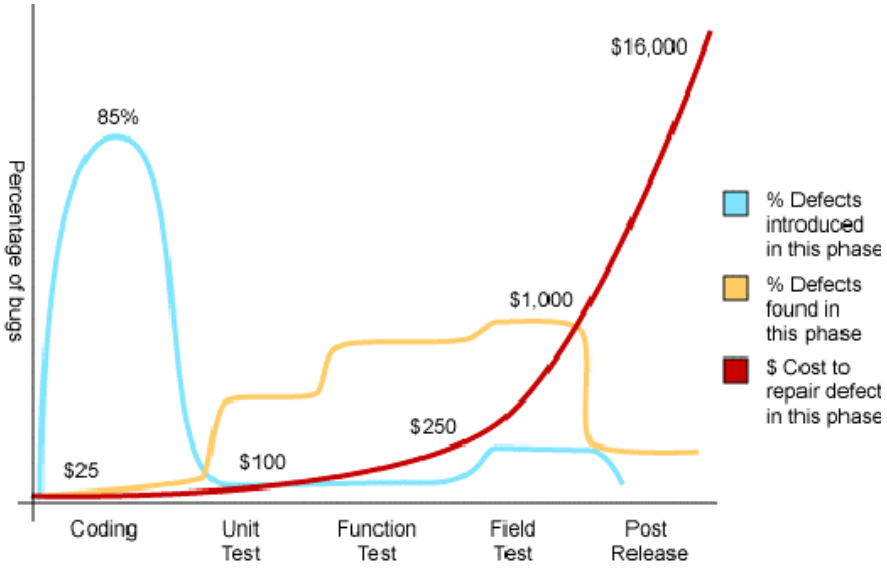
Software Security Costs:

» Unbudgeted time to fix security problems:1000 man-hours (Microsoft)

» Cost of training software developers in security: $100 Million (Microsoft)

» Inadequate software testing costs: $3.3 billion (NIST)

When we do address the problem?



Source: Applied Software Measurement, Capers Jones, 1996

» Today most people test after software is built!

When is more cost effective to build security in?

» Assume the following data from a study (IBM):

- Secure Software Engineering Expense Per Phase
- Number of Security Defects found Per Phase
- Percentage of Vulnerabilities Fixed

» The Return Of Security Investment (ROSI) in dollar savings **for every $ 100,000 spent is:**

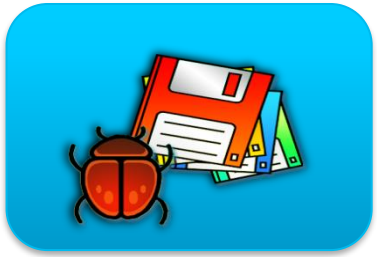
- $ 21,000 when defects are fixed and identified during design
- $ 15,000 when defects are fixed during implementation
- $ 12,000 when defects are fixed during tests

S-SDLC

Software Risk Management and Secure Software Development Life Cycles (S-SDLC)

Cybercrime Evolution

1986-1995



- LANs
- First PC virus
- Motivation: damage

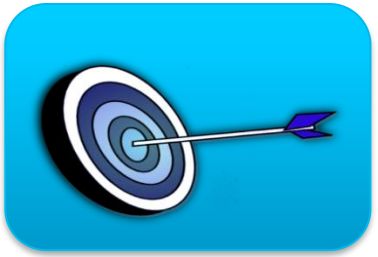
1995-2003

- Internet Era
- "Big Worms"
- Motivation: damage

2004+

- OS, DB attacks
- Spyware, Spam
- Motivation: Financial

2006+



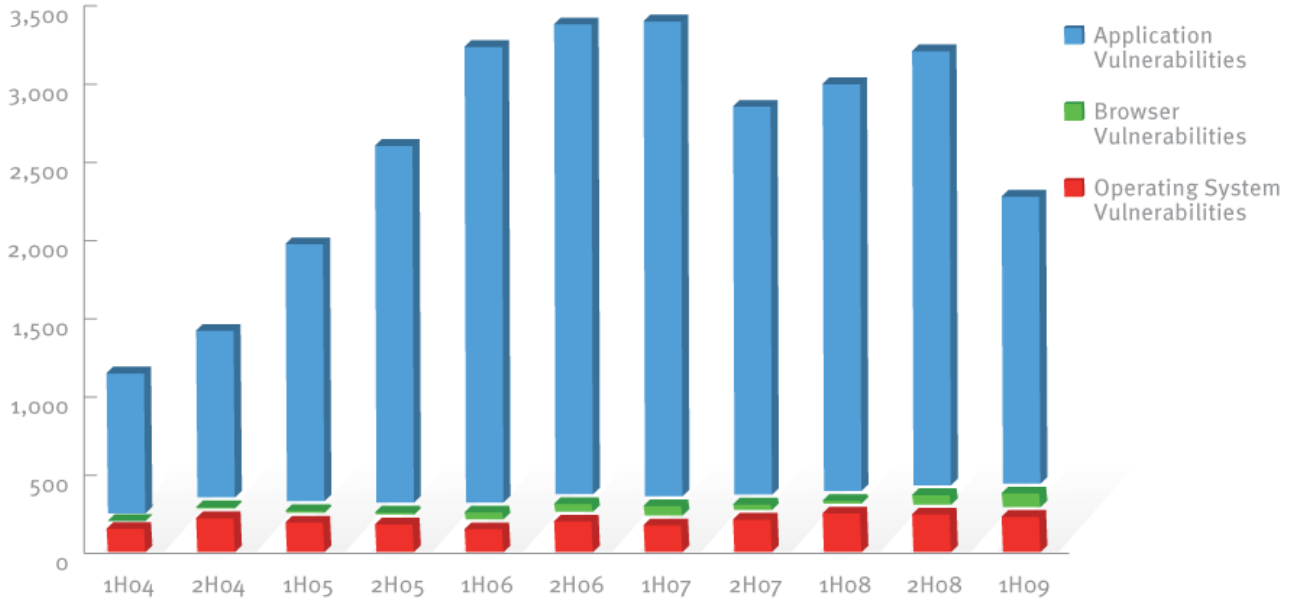
- Targeted attacks
- Social engineering
- Financial + Political

→ ***Cost of U.S. cybercrime: About $70B***

2007 Market prices:	
Credit Card Number	$0.50 - $20
Full Identity	$1 - $15
Bank Account	$10 - $1000

Attacks are focusing on applications

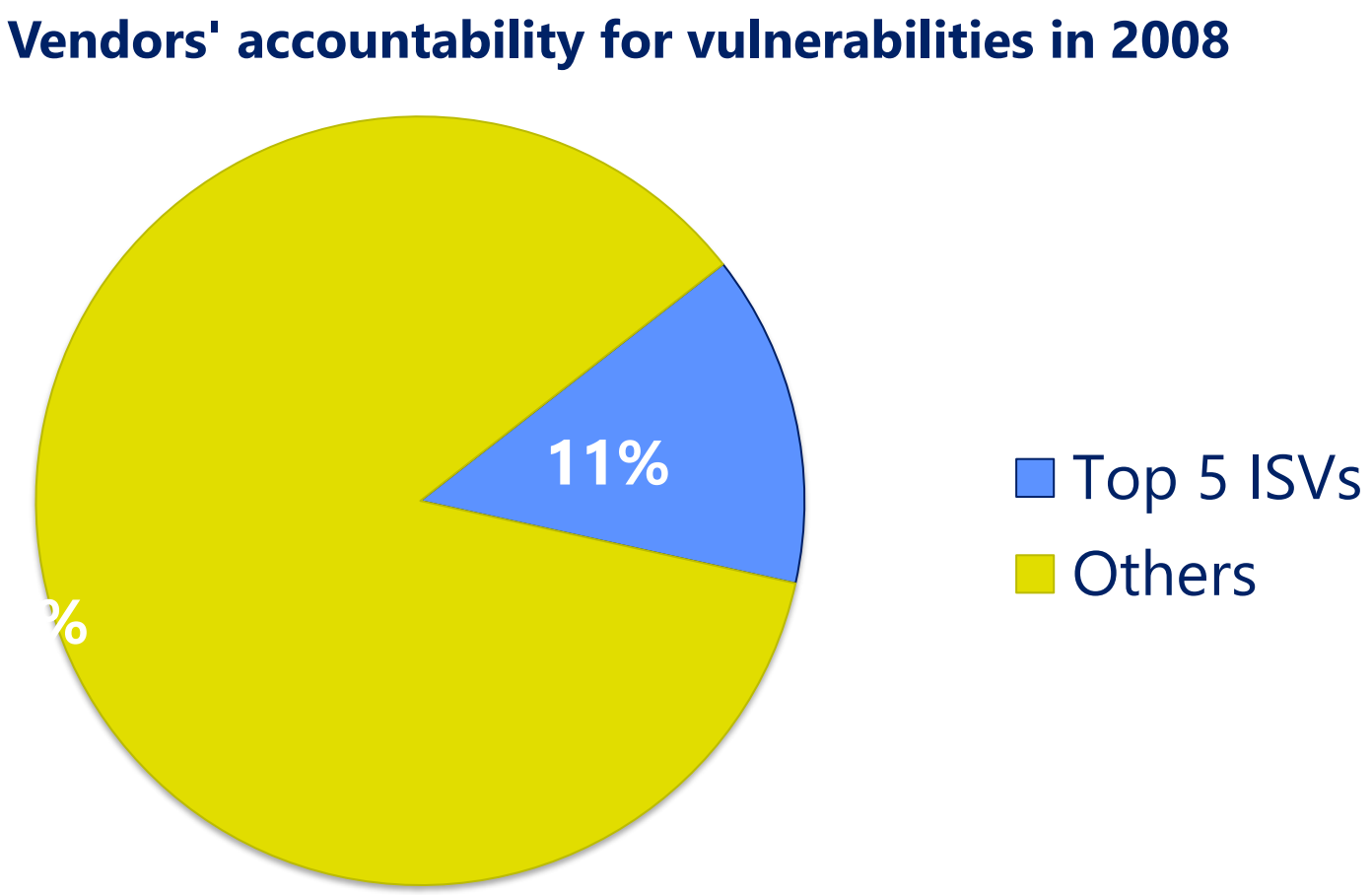
% of vulnerability disclosures:
Operating system vs browser and application vulnerabilities



From the Microsoft Security Intelligence Report V7

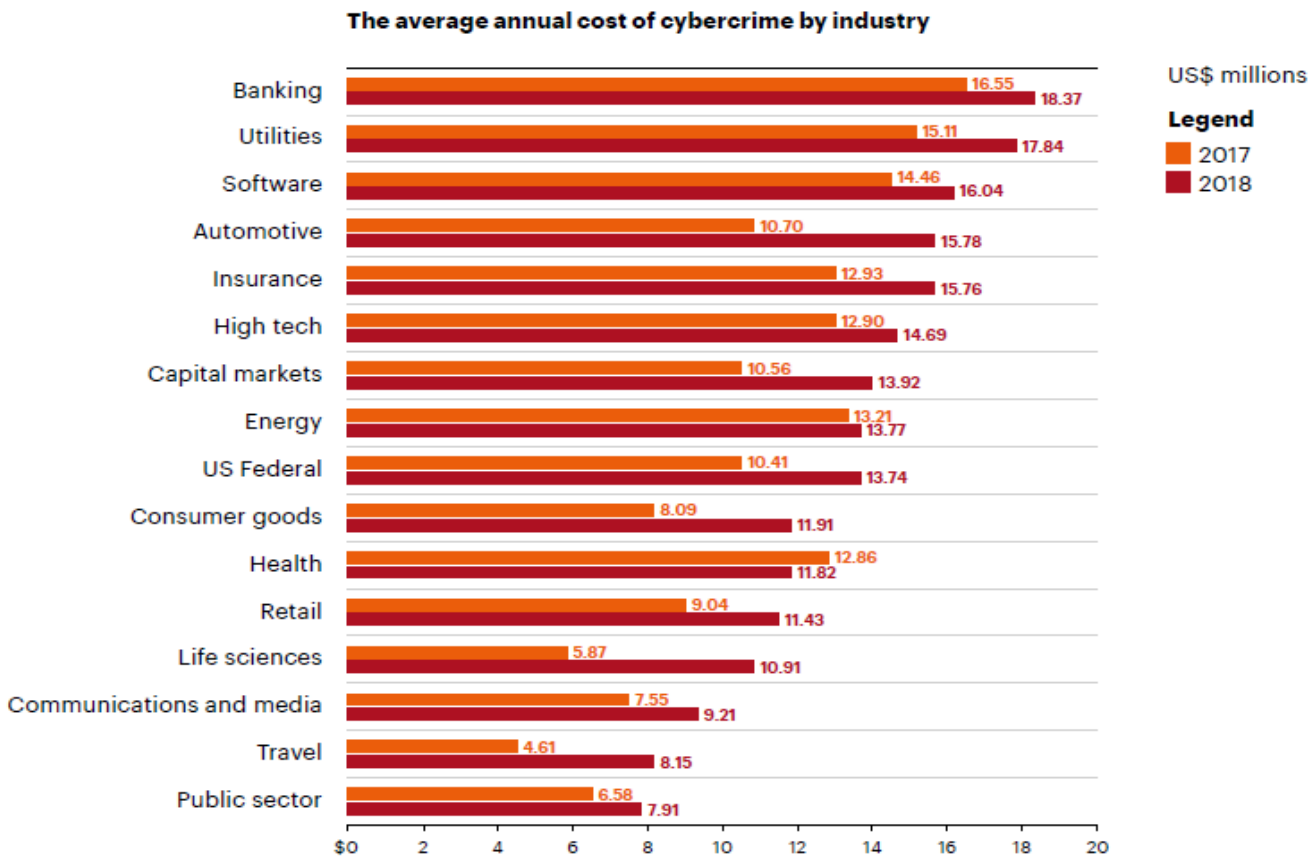
90% of vulnerabilities are remotely exploitable

Most vulnerabilities are in smaller ISV apps

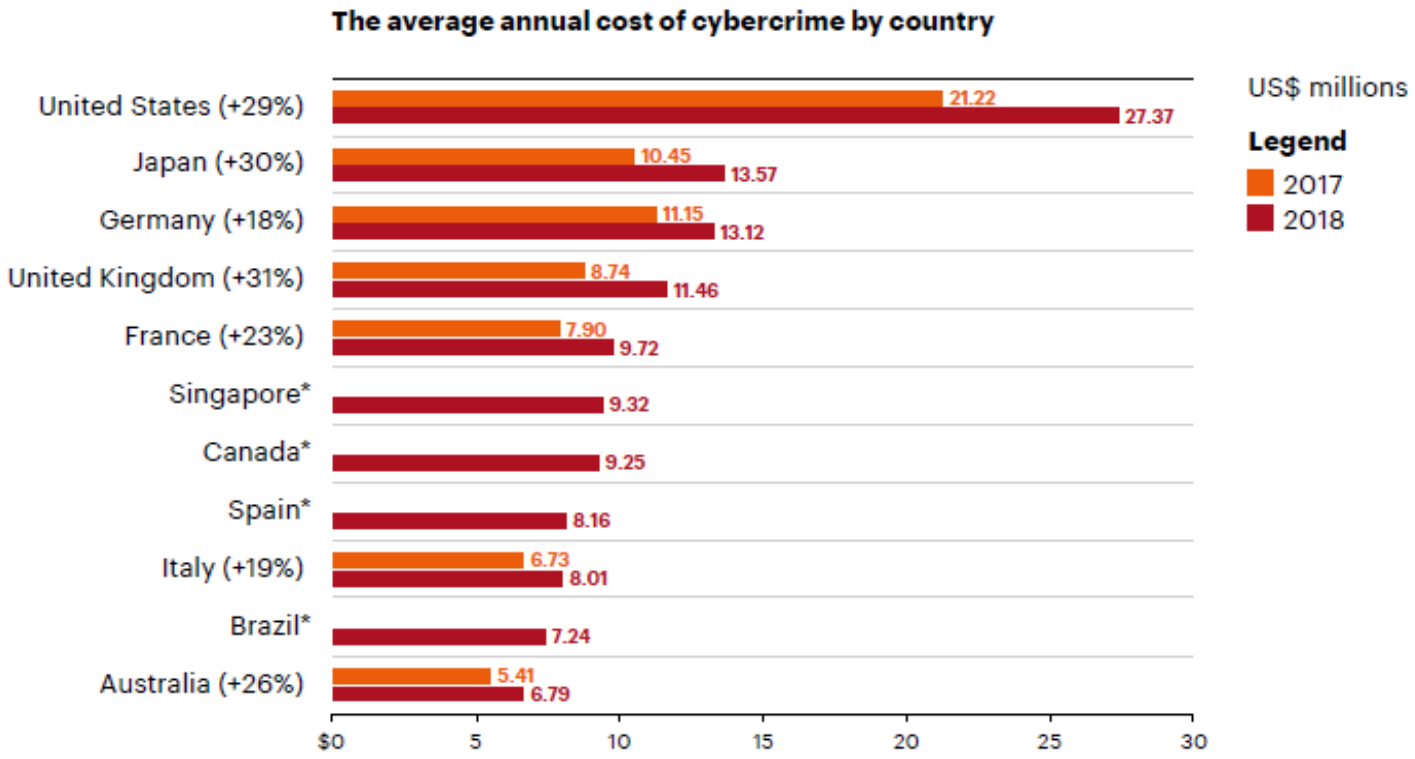


BENCHMARKING CYBERSECURITY INVESTMENT

BENCHMARKING CYBERSECURITY INVESTMENT



BENCHMARKING CYBERSECURITY INVESTMENT



BENCHMARKING CYBERSECURITY INVESTMENT

Consolidating these findings across industries globally, we found that the total value at risk from cybercrime is US$5.2 trillion over the next five years (see Figure 5).

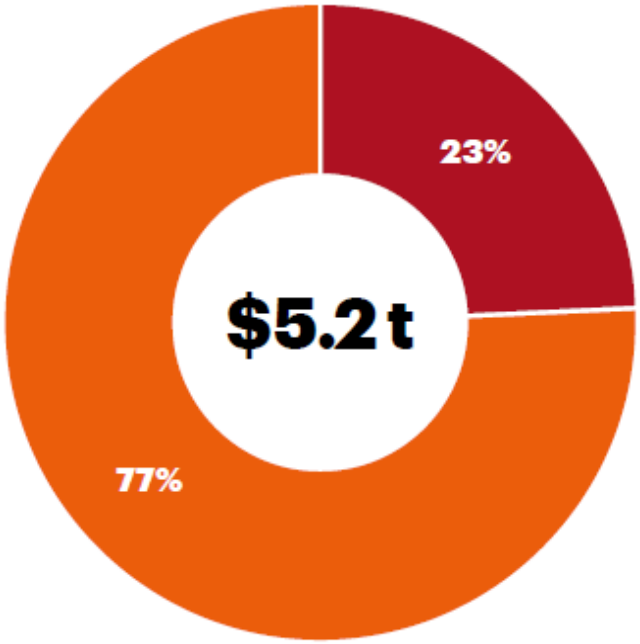
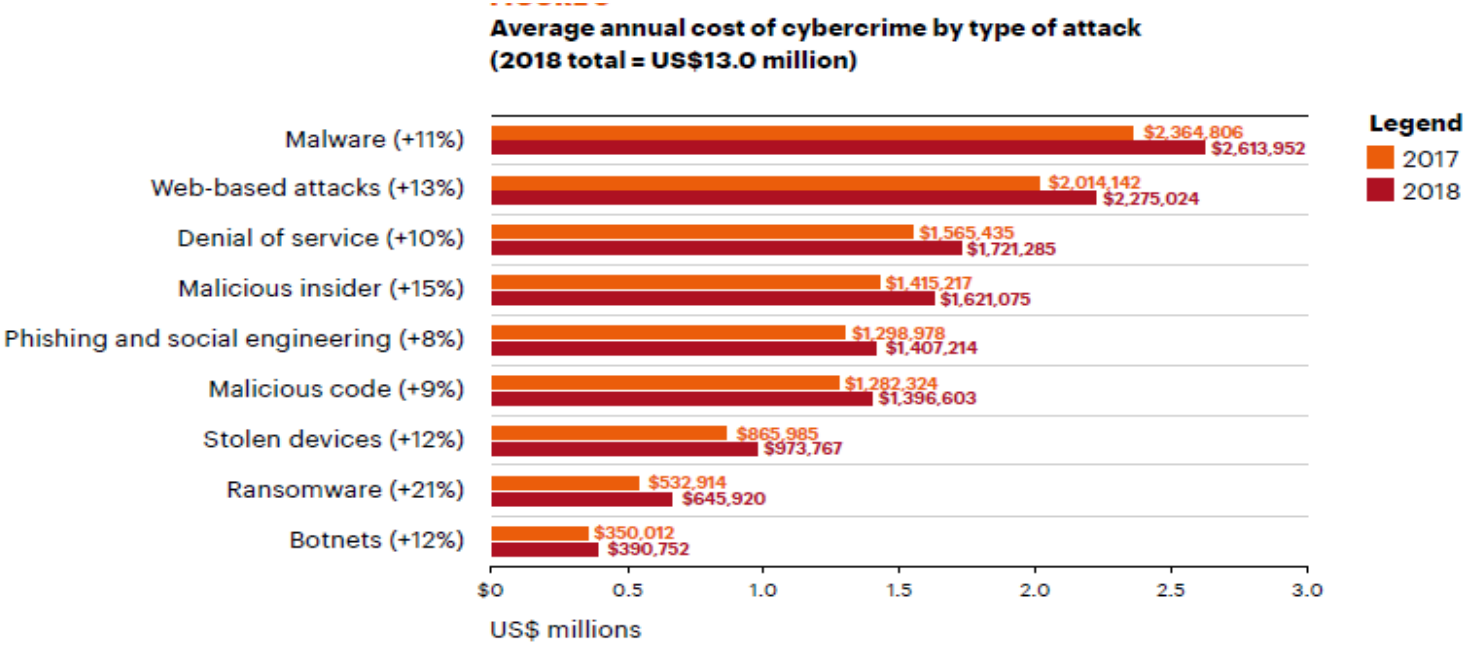


FIGURE 5
Value at risk globally from direct and indirect cyberattacks (Cumulative 2019 to 2023)

Legend
- Value at risk from direct attacks
- Value at risk from indirect attacks

BENCHMARKING CYBERSECURITY INVESTMENT

Malware is the most expensive attack type for organizations. The cost of malware attacks has increased by 11% over the year, and the cost of malicious insider attacks has increased by 15%.

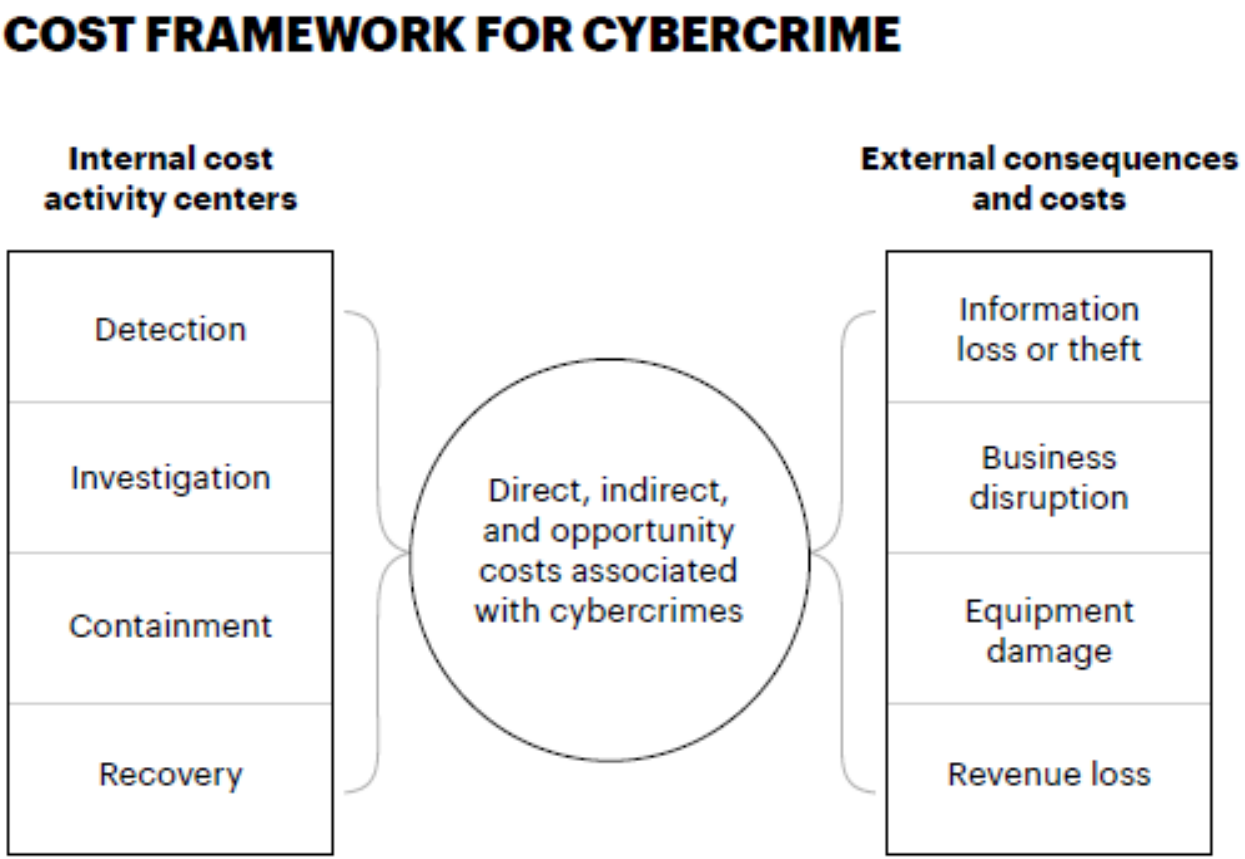


BENCHMARKING CYBERSECURITY INVESTMENT

Consequences of different types of cyberattacks
(average annual cost; figures in US$ million; 2018 total = US$13.0 million)

	Business disruption	Information loss	Revenue loss	Equipment damage	Total cost by attack type
Malware (+11%)	$ 0.5	$ 1.4	$ 0.6	$ 0.1	$ 2.6
Web-based attacks (+17%)	$ 0.3	$ 1.4	$ 0.6	$ –	$ 2.3
Denial-of-service (+10%)	$ 1.1	$ 0.2	$ 0.4	$ 0.1	$ 1.7
Malicious insiders (+15%)	$ 0.6	$ 0.6	$ 0.3	$ 0.1	$ 1.6
Phishing and social engineering (+8%)	$ 0.4	$ 0.7	$ 0.3	$ –	$ 1.4
Malicious code (+9%)	$ 0.2	$ 0.9	$ 0.2	$ –	$ 1.4
Stolen devices (+12%)	$ 0.4	$ 0.4	$ 0.1	$ 0.1	$ 1.0
Ransomware (+21%)	$ 0.2	$ 0.3	$ 0.1	$ 0.1	$ 0.7
Botnets (+12%)	$ 0.1	$ 0.2	$ 0.1	$ –	$ 0.4
Total cost by consequence	$ 4.0	$ 5.9	$ 2.6	$ 0.5	**$ 13.0**

BENCHMARKING CYBERSECURITY INVESTMENT



BENCHMARKING CYBERSECURITY INVESTMENT

- Direct cost—the direct expense outlay to accomplish a given activity.
- Indirect cost—the amount of time, effort and other organizational resources spent, but not as a direct cash outlay.
- Opportunity cost—the cost resulting from lost business opportunities as a consequence of reputation diminishment after the incident.

Security Incidents

- Security incidents indicate the failure of security measures or the breach of organizations' systems or data.
- This includes any event that threatens the integrity, availability, or confidentiality of information.
- Causes of security incidents include perimeter breaches, cyber attacks, and insider threats.

Notable Security Breaches

- **Home Depot data breach**—the largest credit-card compromise and point-of-sale in history. This breach affected 56 million credit cards and 53 million email addresses. For almost six months in 2014, attackers lifted information from the PoS terminals at self-check-out lanes using custom-built malware. Home Depot was forced to **pay $27.25 million in compensation to financial institutions**.
- **Morrisons**—this British supermarket chain was compromised in 2014. A disgruntled employee leaked the entire employee database online, exposing information about over 100,000 employees. Over 2000 of these employees filed a class action lawsuit**, damaging the reputation of the company**. This breach highlights the risk of insider threats.

Notable Security Breaches

- **Yahoo breaches of 2013 and 2014**—a series of major data breaches compromised the accounts of over 1 billion users in 2013 and over 500 million in 2014. Anonymous hackers stole names, phone numbers, passwords, and email addresses. The breaches were only exposed in 2016, making them harder to investigate. The late discovery resulted in greater damage and higher remediation costs.

Types of Security Incidents

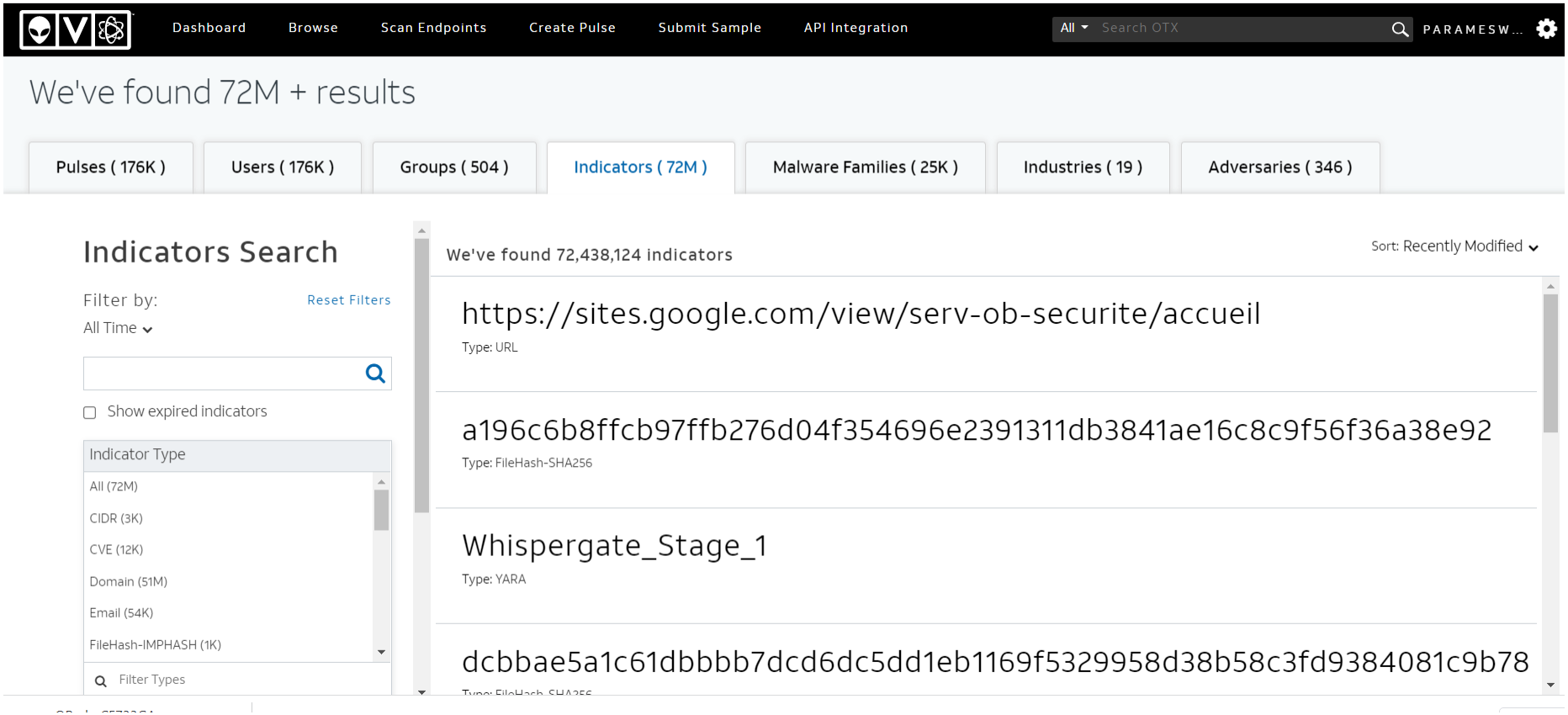
- Brute force attacks
- Email
- Web
- Loss or theft of equipment

Automated Incident Response

- Security Orchestration, Automation and Response (SOAR)
- A SOAR system can:
  - Collect security threat data and alerts
  - Define and enforce a standard workflow for IR activities
  - Analyze incidents, including triage and prioritization
  - Enable automated security playbooks which encode incident analysis and response into a standard, fully-automated or semi-automated process.

Security Incident Tools

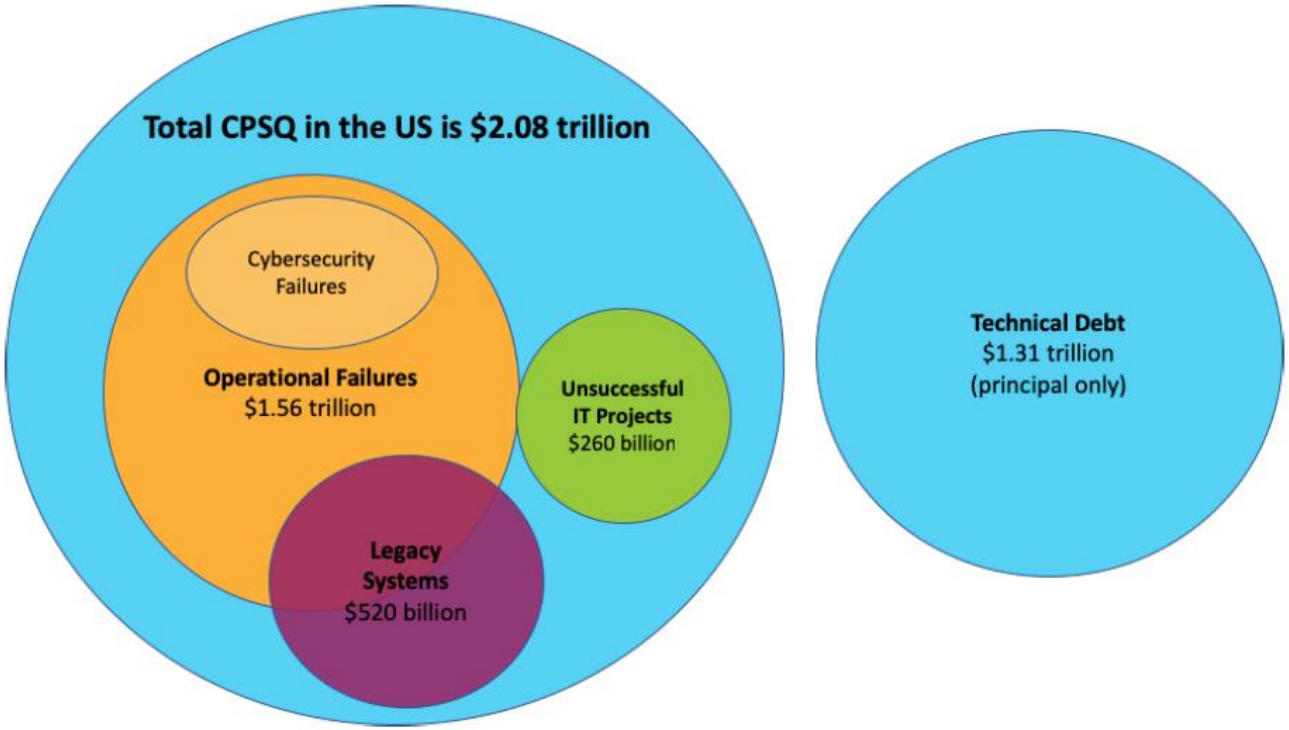
- IBM QRADAR, ALIEN VAULT, R7 Rapid etc.,



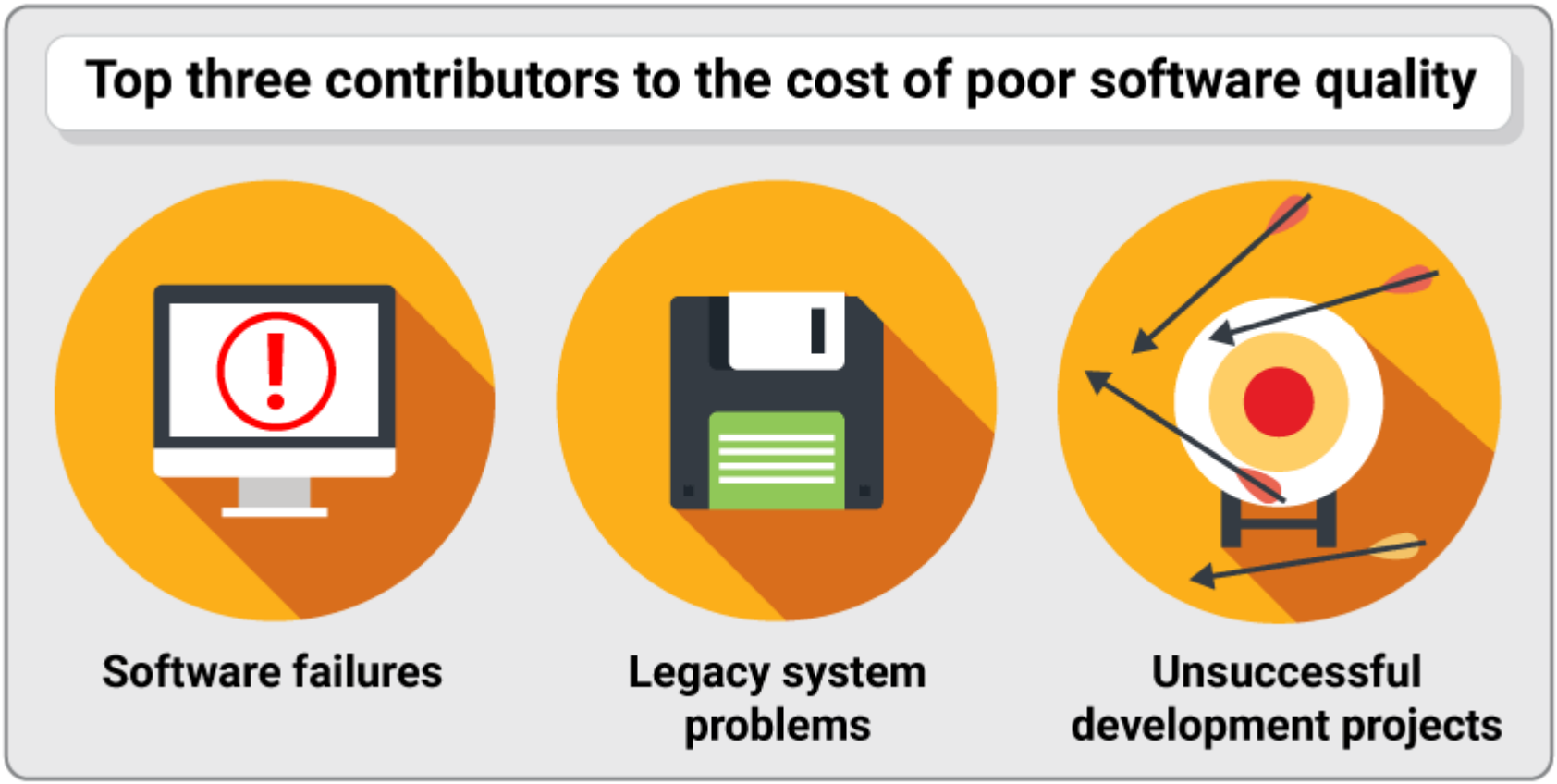
Total Cost of Poor Software Quality (CPSQ)

- The large majority (75%, or an estimated $1.56 trillion) of the CPSQ is software failure caused by the failure to patch known vulnerabilities. That's up 22% from 2018.
- The second-largest piece of the overall cost, $520 billion, is legacy system problems, although that has declined from $635 billion two years ago.
- In third place is unsuccessful development projects, which cost an estimated $260 billion.
- And that number indicates an ominous trend—it's up 46% since 2018.
- The 2020 report observes that while there are multiple factors causing project failures, "one consistent theme has been the lack of attention to quality."

Total Cost of Poor Software Quality (CPSQ)

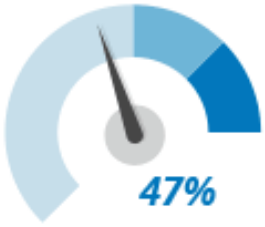


Total Cost of Poor Software Quality (CPSQ)



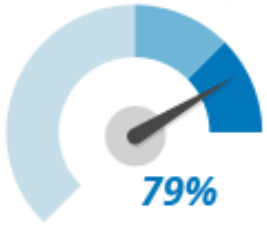
Total Cost of Poor Software Quality (CPSQ)

How website performance affects users behaviour

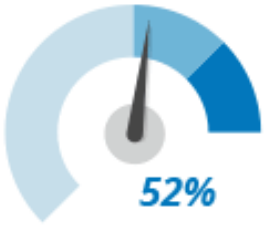


47% of consumers expect a web page to load in 2 seconds or less.

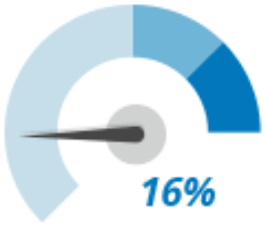
40% abandon a website that takes more than 3 seconds to load.



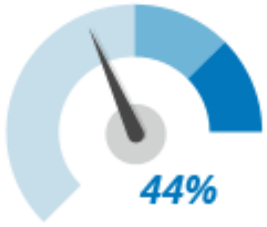
79% of users who are dissatisfied with website performance are less likely to return.



52% of users state that quick page loading is important to their site loyalty.



A 1 second delay (or 3 seconds of waiting) decreases customer satisfaction by about **16%**.



44% of users will tell their friends about a bad experience online.

Need for Secure Coding

- Secure coding is the practice of writing software that's protected from vulnerabilities.

Web Hacking

What is Web application ?

- Unlike other computer based application that runs in the local Operating System of the device, Web application is an application program that runs on the remote server and delivered over the internet through the user's browser interface.

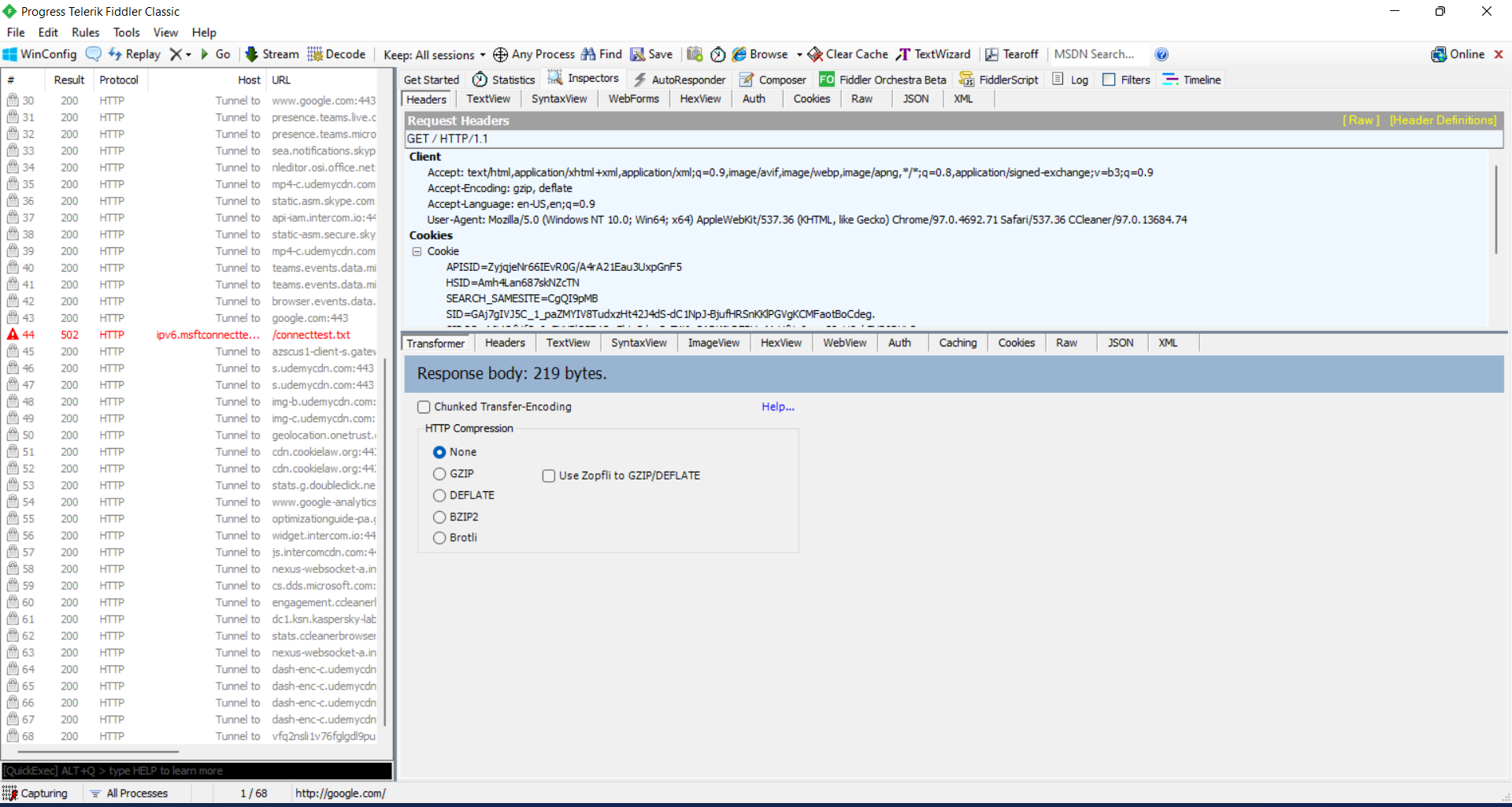
- Client – Server Architecture

- It consist of at least 3 main components : Web Server, Application Server and Database Server

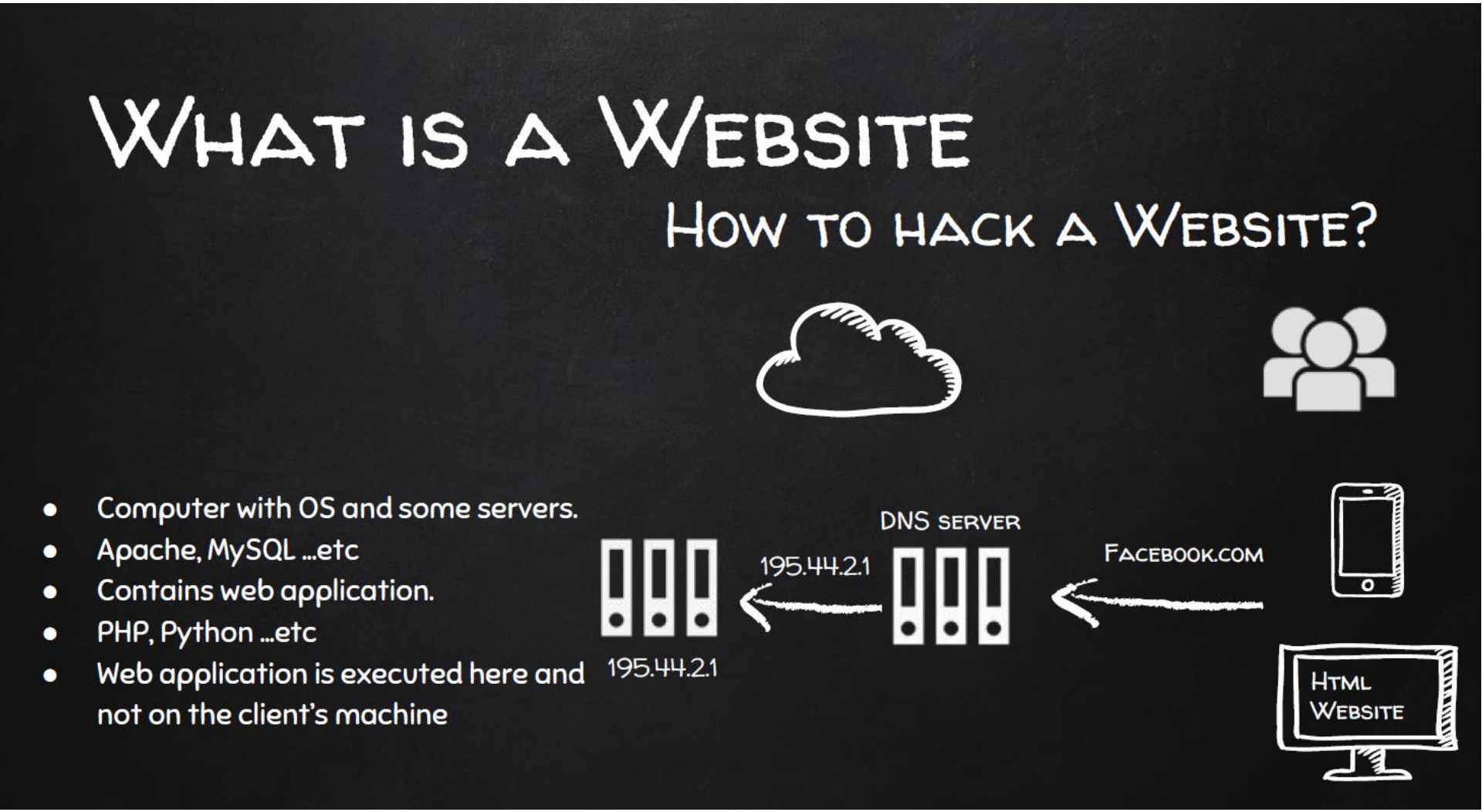
- Browser interacts with the application with the HTTP Request and get the response page as HTTP reply.

- HTTP is the protocol that governs the communication between browser and web server.

Web Hacking


Progress Telerik Fiddler Classic
File Edit Rules Tools View Help
WinConfig Replay Go Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard Tearoff MSDN Search... Online
Result Protocol Host URL
Get Started Statistics Inspectors AutoResponder Composer Fiddler Orchestra Beta FiddlerScript Log Filters Timeline
Headers TextView SyntaxView WebForms HexView Auth Cookies Raw JSON XML
Request Headers [Raw] [Header Definitions]
GET / HTTP/1.1
Client
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.71 Safari/537.36 CCleaner/97.0.13684.74
Cookies
Cookie
APISID=ZyjqjeNr66IEvR0G/A4rA21Eau3UxpGnF5
HSID=Amh4Lan687skNZcTN
SEARCH_SAMESITE=CgQI9pMB
SID=GAj7gIVJ5C_1_paZMYIV8TudxzHt42J4dS-dC1NpJ-BjufHRSnKKlPGVgKCMFaotBoCdeg.
Transformer Headers TextView SyntaxView ImageView HexView WebView Auth Caching Cookies Raw JSON XML
Response body: 219 bytes.
Chunked Transfer-Encoding
Help...
HTTP Compression
None
GZIP
DEFLATE
BZIP2
Brotli
Use Zopfli to GZIP/DEFLATE
[QuickExec] ALT+Q > type HELP to learn more
Capturing All Processes 1 / 68 http://google.com/

Web Hacking


What is a Website
How to hack a Website?
Computer with OS and some servers.
Apache, MySQL ...etc
Contains web application.
PHP, Python ...etc
Web application is executed here and not on the client's machine
195.44.2.1
195.44.2.1
DNS server
Facebook.com
Html Website

Web Hacking


What is a Website
How to hack a Website?
An application installed on a computer . → web application pentesting
Computer uses an OS + other applications → server side attacks.
Managed by humans → client side attacks.

Web Hacking

Information Gathering

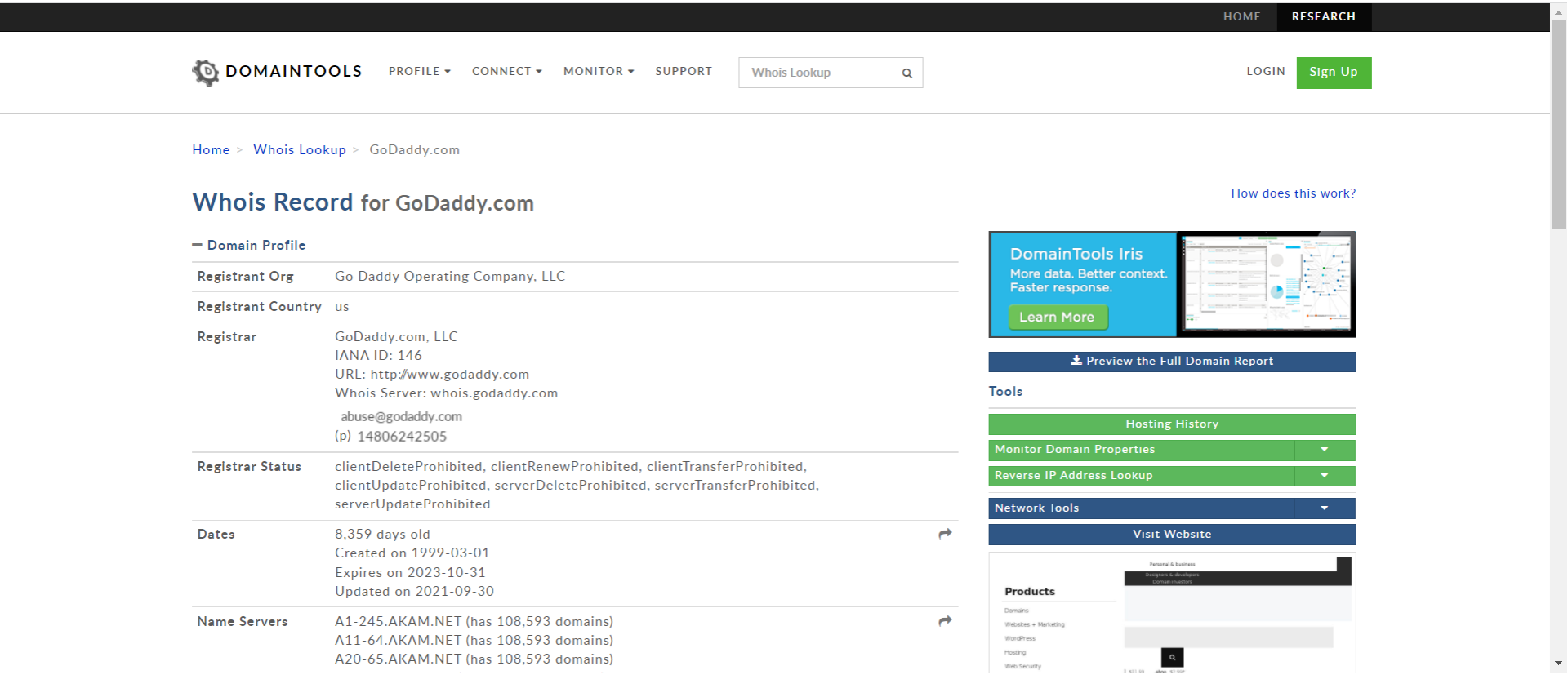
- IP address.
- Domain name info.
- Technologies used.
- Other websites on the same server.
- DNS records.
- Unlisted files, sub-domains, directories.

Information Gathering

1. Whois Lookup – Find info about the owner of the target.
   → http://whois.domaintools.com/
2. Netcraft Site Report – Shows technologies used on the target.
   → http://toolbar.netcraft.com/site_report?url=
3. Robtex DNS lookup – Shows comprehensive info about the target website.
   → https://www.robtex.com/

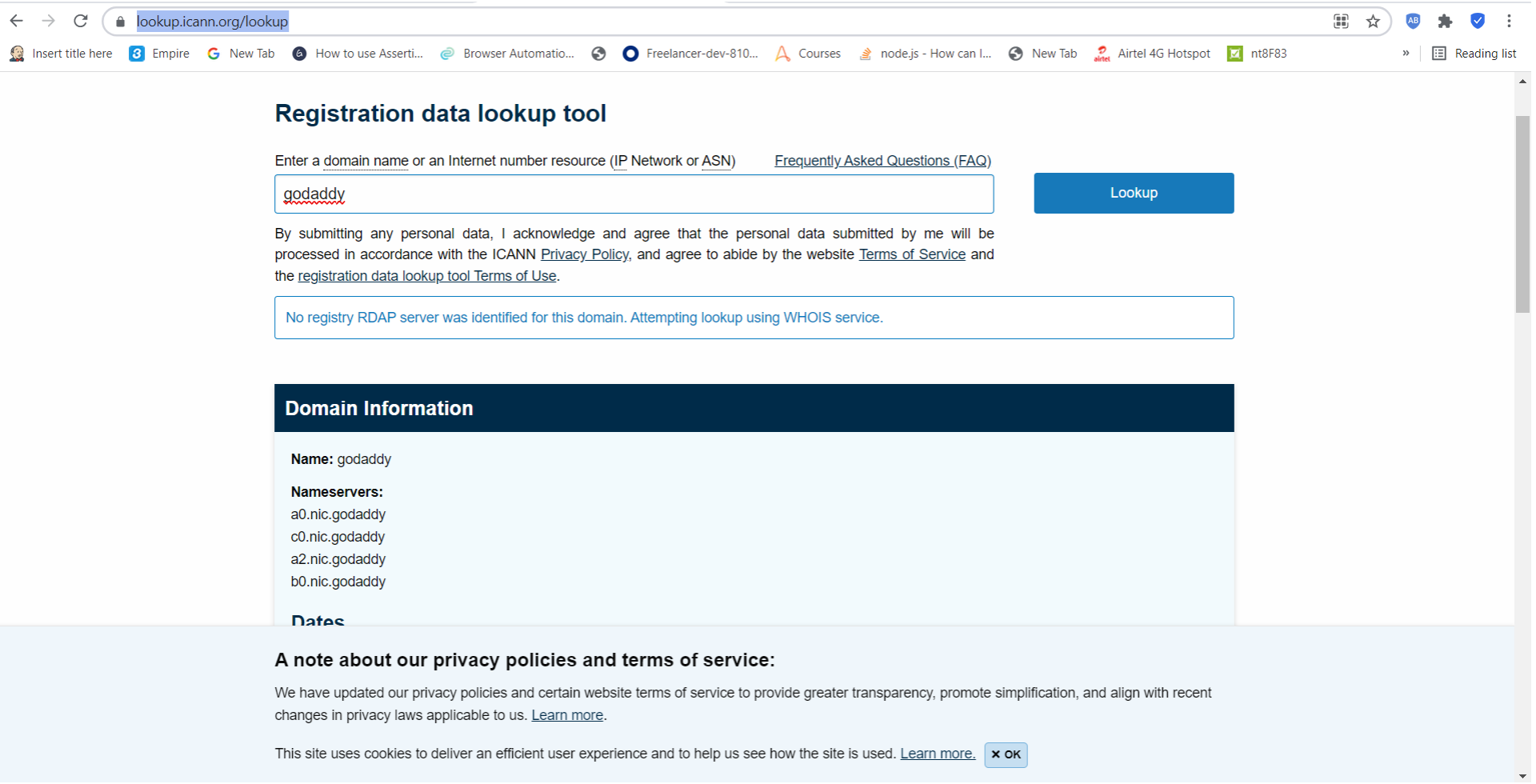
Who is lookup

https://whois.domaintools.com/



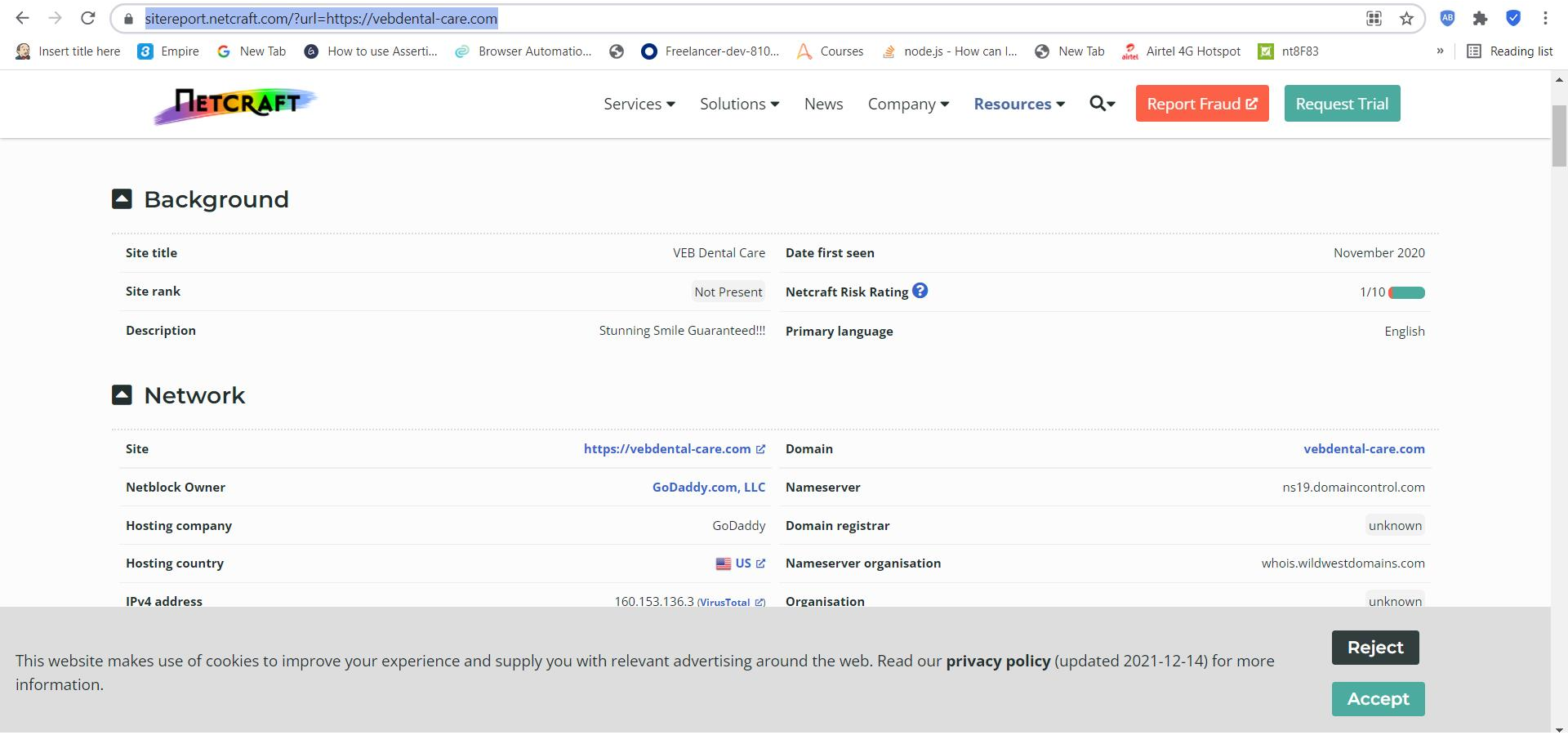
Who is lookup

https://lookup.icann.org/lookup



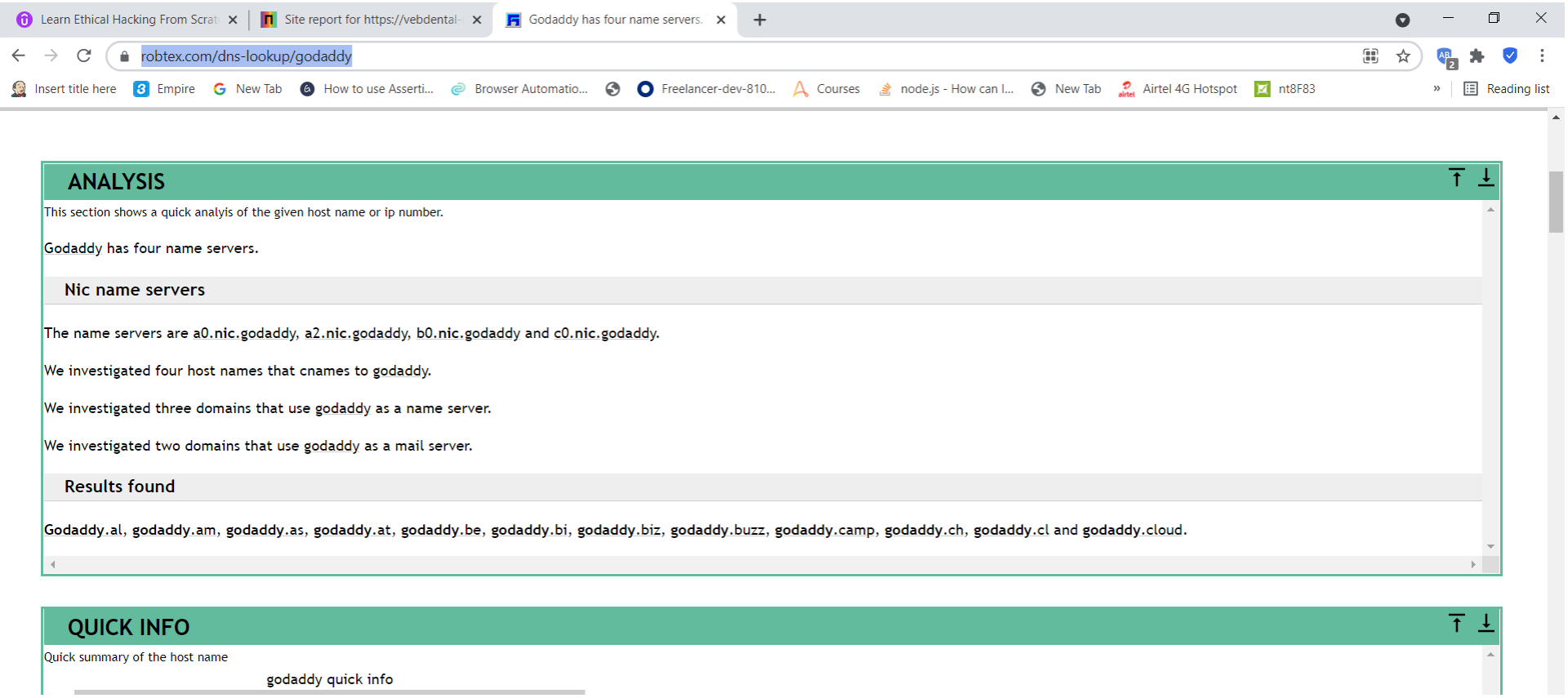
Show Technologies used on the target

- http://toolbar.netcraft.com/site_report?url=https://vebdental-care.com



DNS lookup

- https://www.robtex.com



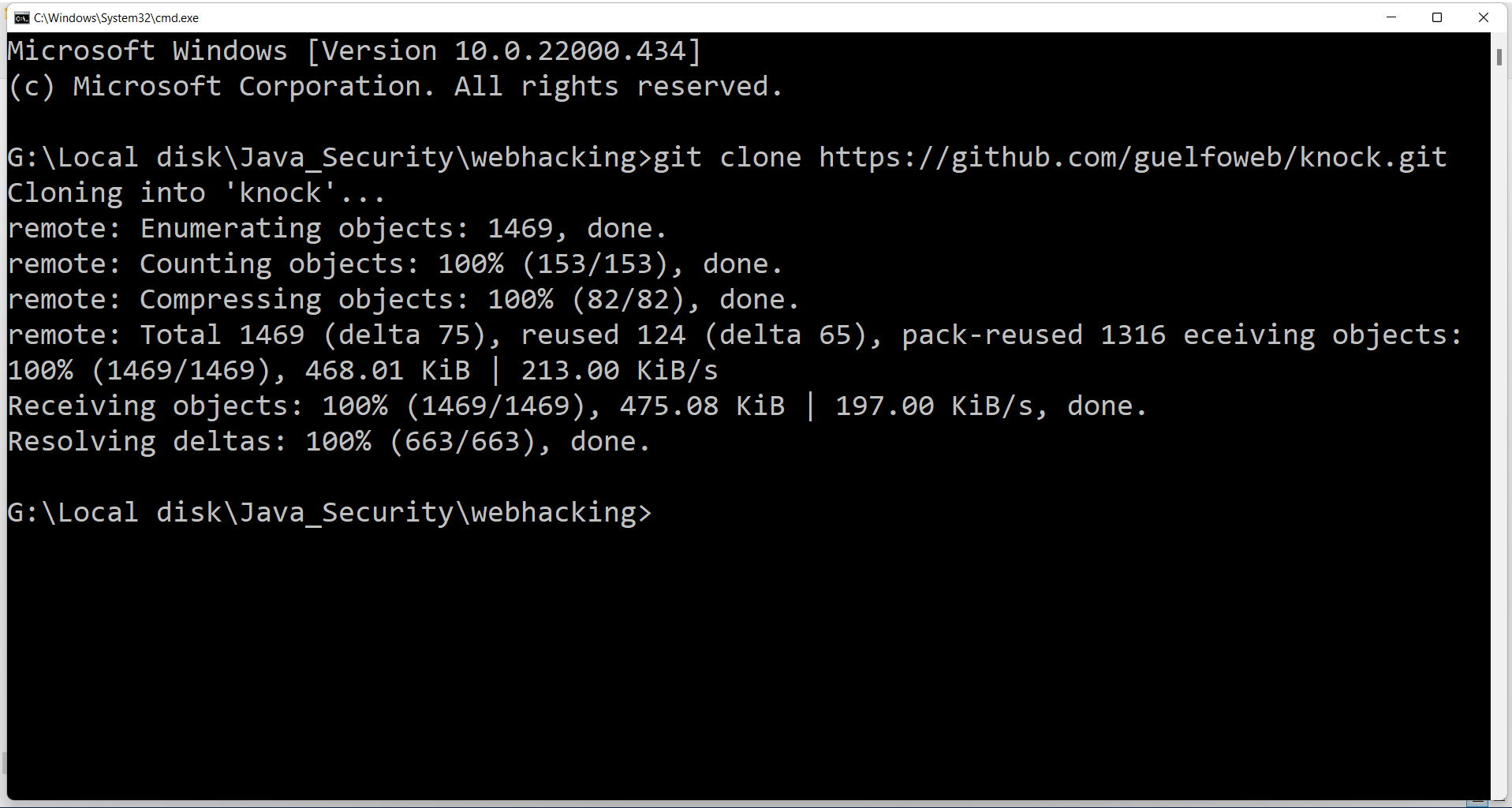
Information Gathering on Sub Domains

- Subdomain.target.com
- Ex: beta.facebook.com

Knock can be used to find subdomains of target

1. Download it > git clone https://github.com/guelfoweb/knock.git
2. Navigate to knock.py. > ce knock/knock.py
3. Run it > python knock.py [target]

Information Gathering on Sub Domains



Information Gathering on Sub Domains

C:\Windows\System32\cmd.exe - python knock/knockpy.py godaddy.com

```
knockpy: error: the following arguments are required: domain

G:\Local disk\Java_Security\webhacking>python knock/knockpy.py https://godaddy.com
remove http(s)://

G:\Local disk\Java_Security\webhacking>python knock/knockpy.py godaddy.com


  _  __                  _
 | |/ /                 | |   v5.2.0
 | ' / _ __   ___   ___| | ___ __  _   _
 |  < | '_ \ / _ \ / __| |/ / '_ \| | | |
 | . \| | | | (_) | (__|   <| |_) | |_| |
 |_|\_\_| |_|\___/ \___|_|\_\ .__/ \__, |
                            | |     __/ |
                            |_|    |___/

←[1m←[33mlocal: ←[0m←[1m←[36m10757←[0m ←[35m| ←[0m←[2mgoogle: ←[0m←[2m0←[0m ←[35m| ←[0m←[1
m←[33mduckduckgo: ←[0m←[1m←[36m9←[0m ←[35m| ←[0m←[2mvirustotal: ←[0m←[2m0←[0m ←[35m
←[0m
←[1m←[33mWordlist: ←[0m←[1m←[36m10766←[0m ←[35m| ←[0m←[1m←[33mTarget: ←[0m←[1m←[36mgodaddy
.com←[0m ←[35m| ←[0m←[1m←[33mIp: ←[0m←[1m←[36m208.109.192.70←[0m ←[35m
←[0m
```

Information Gathering on Sub Domains

C:\Windows\System32\cmd.exe

```
66.210.39.2          vpn.godaddy.com

184.168.130.180      vortex.godaddy.com

63.241.136.27        webboard.godaddy.com

64.202.166.117       webmail1.godaddy.com

184.168.130.29       webext.godaddy.com

63.241.136.95        webservices.godaddy.com

66.210.39.2          webvpn.godaddy.com

63.241.136.168       www1.godaddy.com

63.241.136.91        www4.godaddy.com

63.241.136.90        www5.godaddy.com

63.241.136.86        www6.godaddy.com
```

Information Gathering Files and Gathering

- Find files & directories in target website
- A tool called drib.

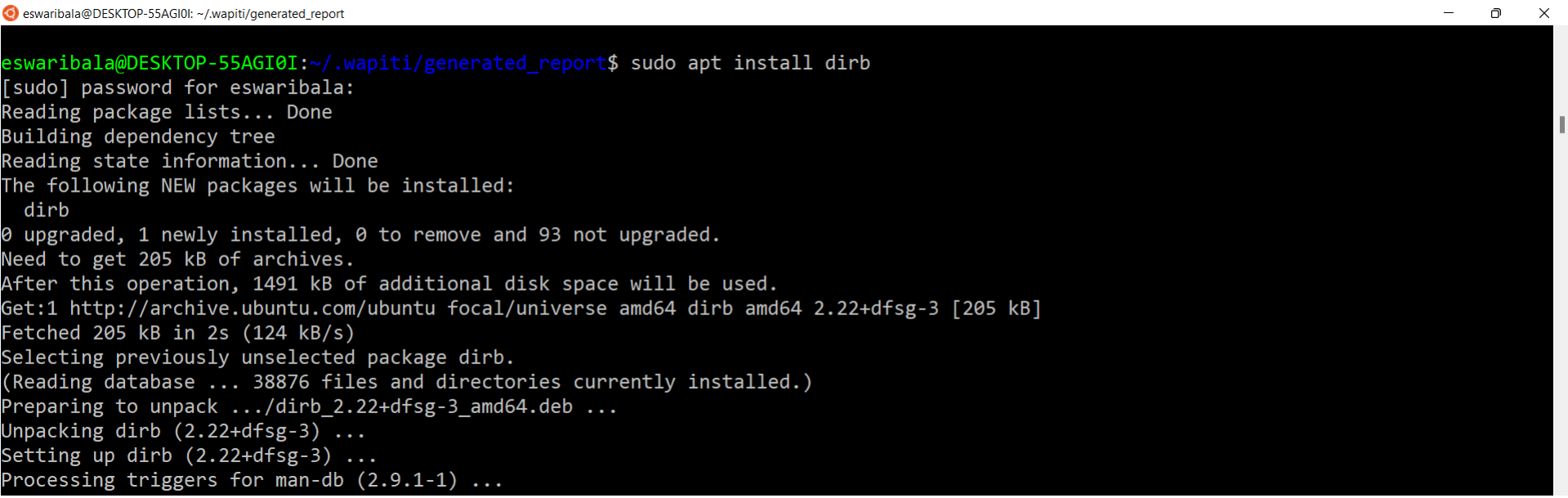
```
> dirb [target] [wordlist] [options]
```

For more info run

```
> man dirb
```

Information Gathering Files and Gathering

- **dirb** http://webscantest.com



Information Gathering Files and Gathering

- **dirb http://webscantest.com**

eswaribala@DESKTOP-55AGI0I: ~/.wapiti/generated_report

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  dirb
0 upgraded, 1 newly installed, 0 to remove and 93 not upgraded.
Need to get 205 kB of archives.
After this operation, 1491 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 dirb amd64 2.22+dfsg-3 [205 kB]
Fetched 205 kB in 2s (124 kB/s)
Selecting previously unselected package dirb.
(Reading database ... 38876 files and directories currently installed.)
Preparing to unpack .../dirb_2.22+dfsg-3_amd64.deb ...
Unpacking dirb (2.22+dfsg-3) ...
Setting up dirb (2.22+dfsg-3) ...
Processing triggers for man-db (2.9.1-1) ...
eswaribala@DESKTOP-55AGI0I:~/.wapiti/generated_report$ dirb http://webscantest.com

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Wed Jan 19 00:08:54 2022
URL_BASE: http://webscantest.com/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://webscantest.com/ ----
--> Testing: http://webscantest.com/_dummy
```

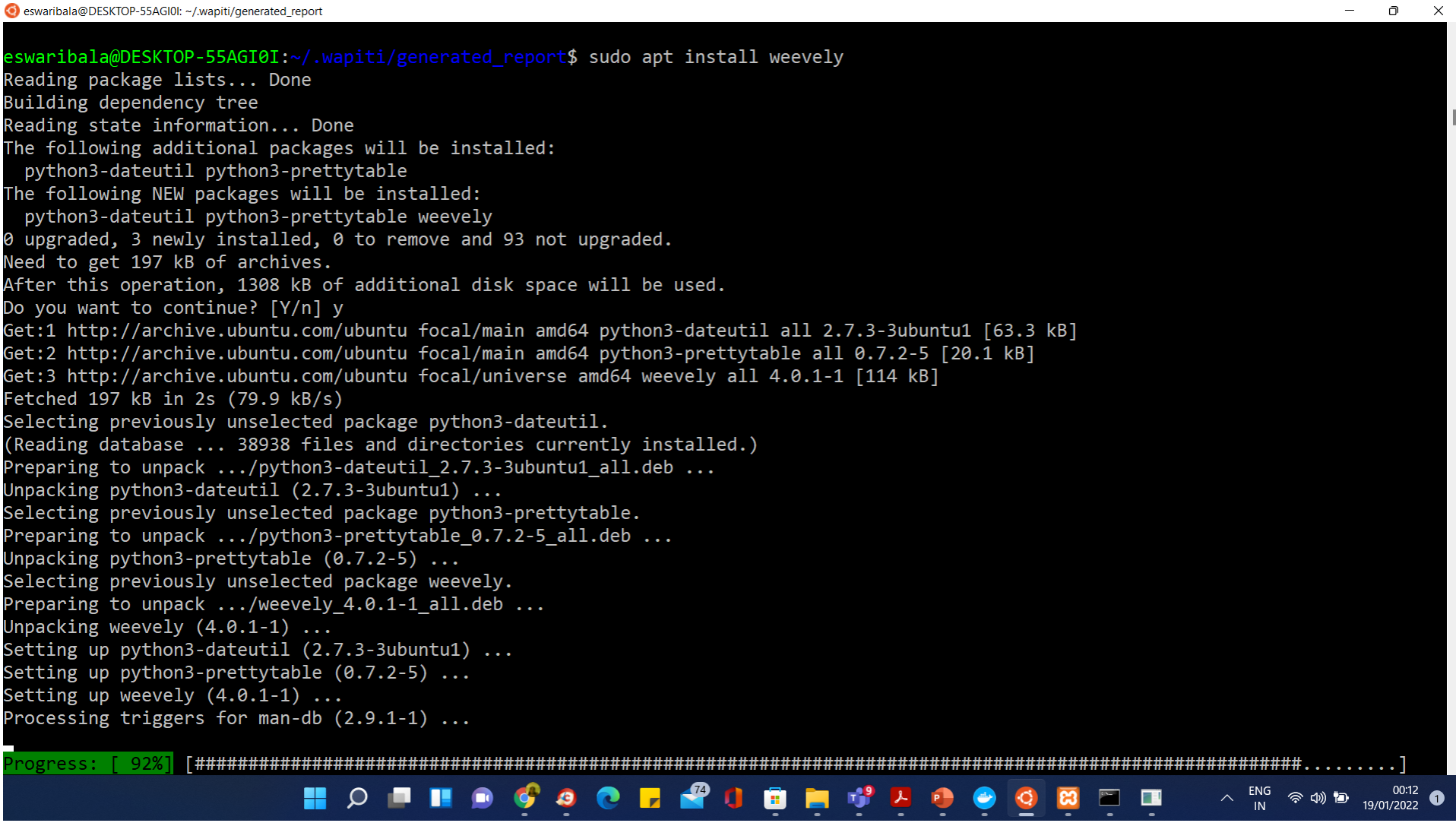
Exploitation File upload Vulnerabilities

- Simples type of vulnerabilities.
- Allow users to upload executable files such as php.

Upload a php shell or backdoor, ex: weevly

1. Generate backdoor > weevly generate [passord] [file name]
2. Upload generated file.
3. Connect to it > weevly [url to file] [password]
4. Find out how to use weevly > help

Weevely


eswaribala@DESKTOP-55AGI0I: ~/.wapiti/generated_report
eswaribala@DESKTOP-55AGI0I:~/.wapiti/generated_report$ sudo apt install weevely
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
python3-dateutil python3-prettytable
The following NEW packages will be installed:
python3-dateutil python3-prettytable weevely
0 upgraded, 3 newly installed, 0 to remove and 93 not upgraded.
Need to get 197 kB of archives.
After this operation, 1308 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-dateutil all 2.7.3-3ubuntu1 [63.3 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-prettytable all 0.7.2-5 [20.1 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal/universe amd64 weevely all 4.0.1-1 [114 kB]
Fetched 197 kB in 2s (79.9 kB/s)
Selecting previously unselected package python3-dateutil.
(Reading database ... 38938 files and directories currently installed.)
Preparing to unpack .../python3-dateutil_2.7.3-3ubuntu1_all.deb ...
Unpacking python3-dateutil (2.7.3-3ubuntu1) ...
Selecting previously unselected package python3-prettytable.
Preparing to unpack .../python3-prettytable_0.7.2-5_all.deb ...
Unpacking python3-prettytable (0.7.2-5) ...
Selecting previously unselected package weevely.
Preparing to unpack .../weevely_4.0.1-1_all.deb ...
Unpacking weevely (4.0.1-1) ...
Setting up python3-dateutil (2.7.3-3ubuntu1) ...
Setting up python3-prettytable (0.7.2-5) ...
Setting up weevely (4.0.1-1) ...
Processing triggers for man-db (2.9.1-1) ...
Progress: [92%] [###.....]
ENG
IN
00:12
19/01/2022

Weevely

```
eswaribala@DESKTOP-55AGI0I:~/.wapiti/generated_report$ weevely generate ddos ddos.php
Generated 'ddos.php' with password 'ddos' of 751 byte size.
eswaribala@DESKTOP-55AGI0I:~/.wapiti/generated_report$ ls
css  ddos.php  logo_clear.png  report.html  whoisdomaintools.com_01182022_1752.html
eswaribala@DESKTOP-55AGI0I:~/.wapiti/generated_report$ weevely help
```

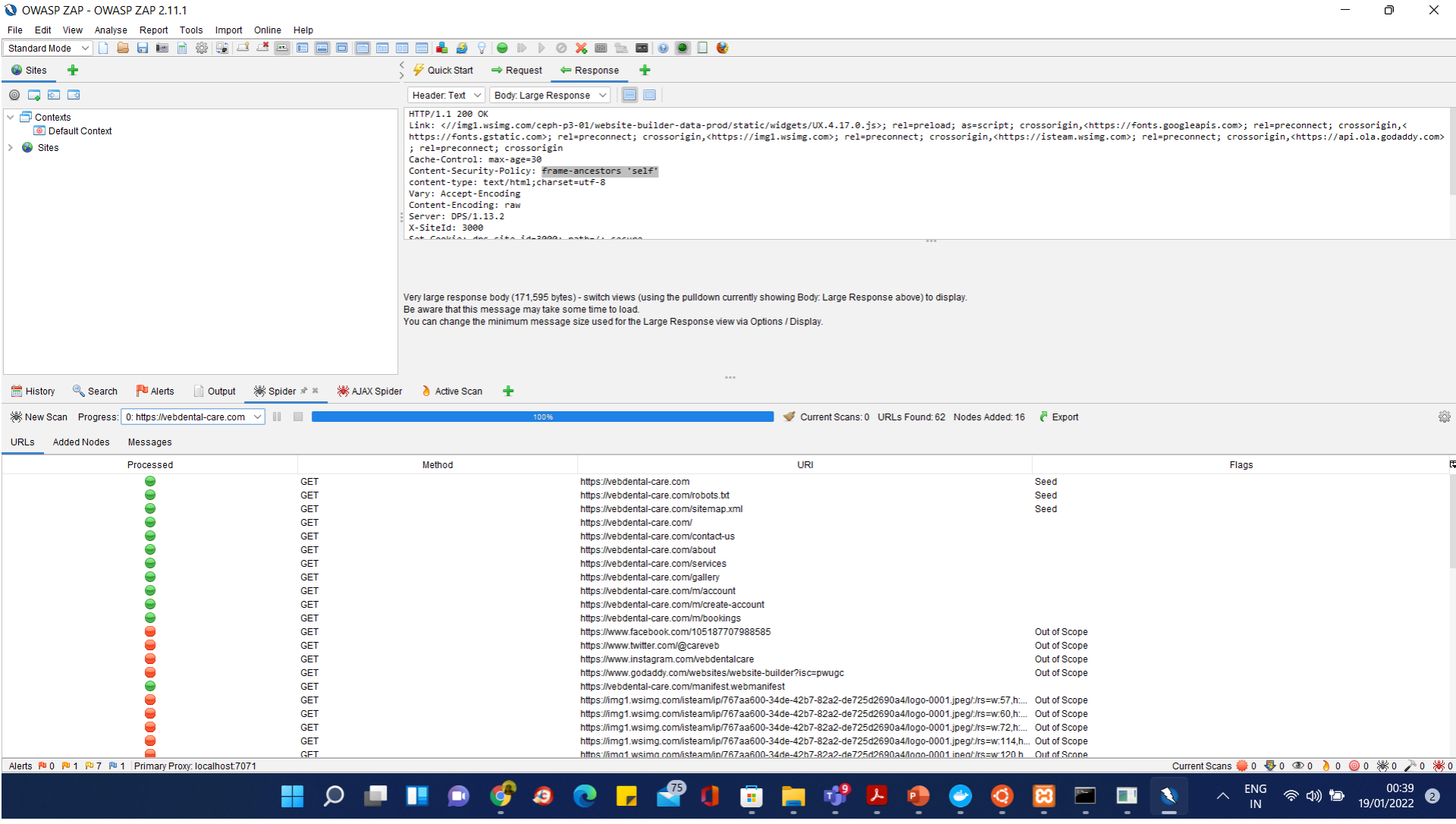
eswaribala@DESKTOP-55AGI0I: ~/.wapiti/generated_report

```
eswaribala@DESKTOP-55AGI0I:~/.wapiti/generated_report$ weevely http://192.168.1.8/ddos.php ddos

[+] weevely 4.0.1

[+] Target:     192.168.1.8
[+] Session:    /home/eswaribala/.weevely/sessions/192.168.1.8/ddos_0.session

[+] Browse the filesystem or execute commands starts the connection
[+] to the target. Type :help for more information.

weevely>
```

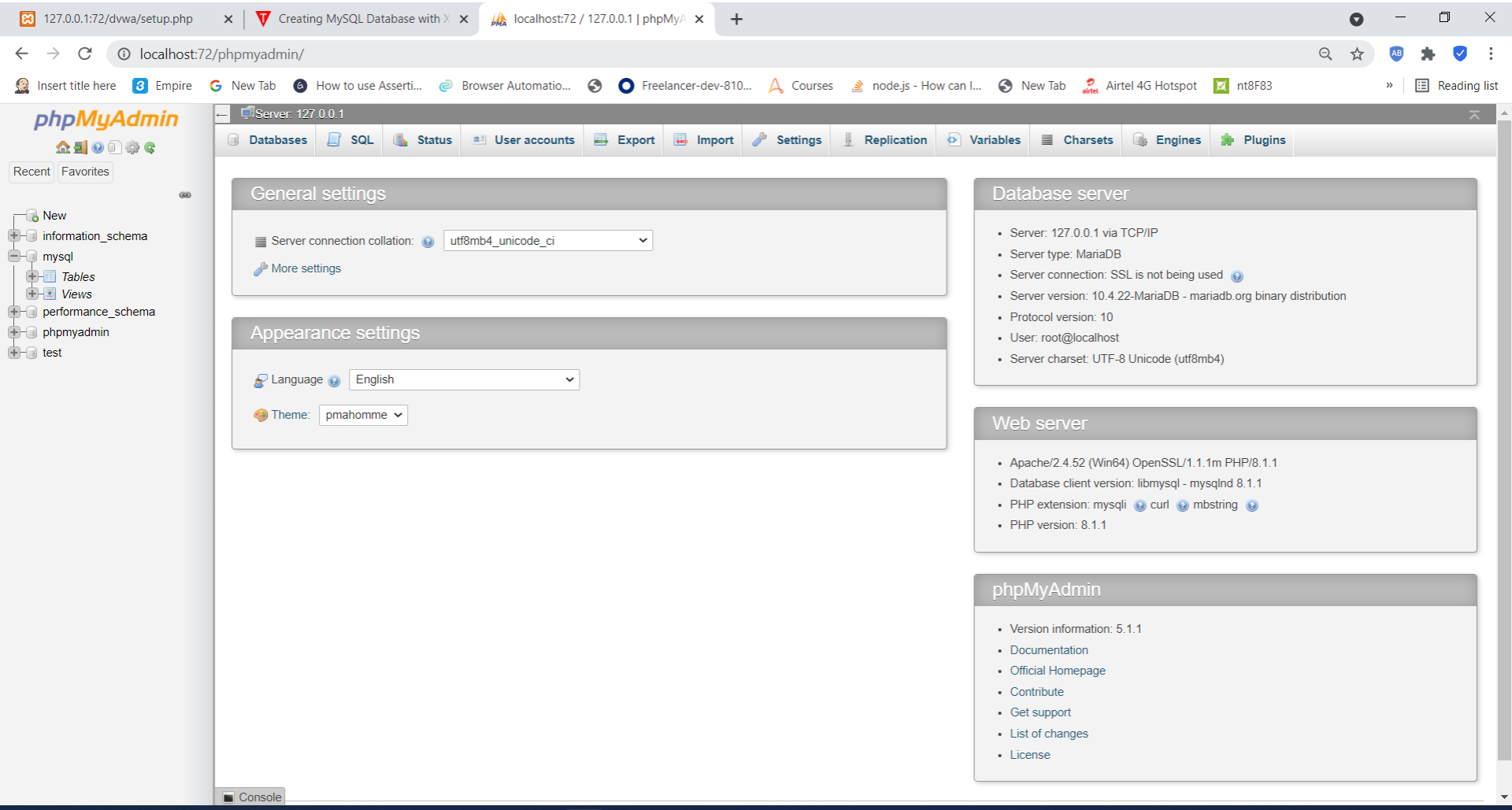
ZED Attack Proxy ZAP

- Automatically find vulnerabilities in web applications.
- Free and easy to use.
- Can also be used for manual testing.

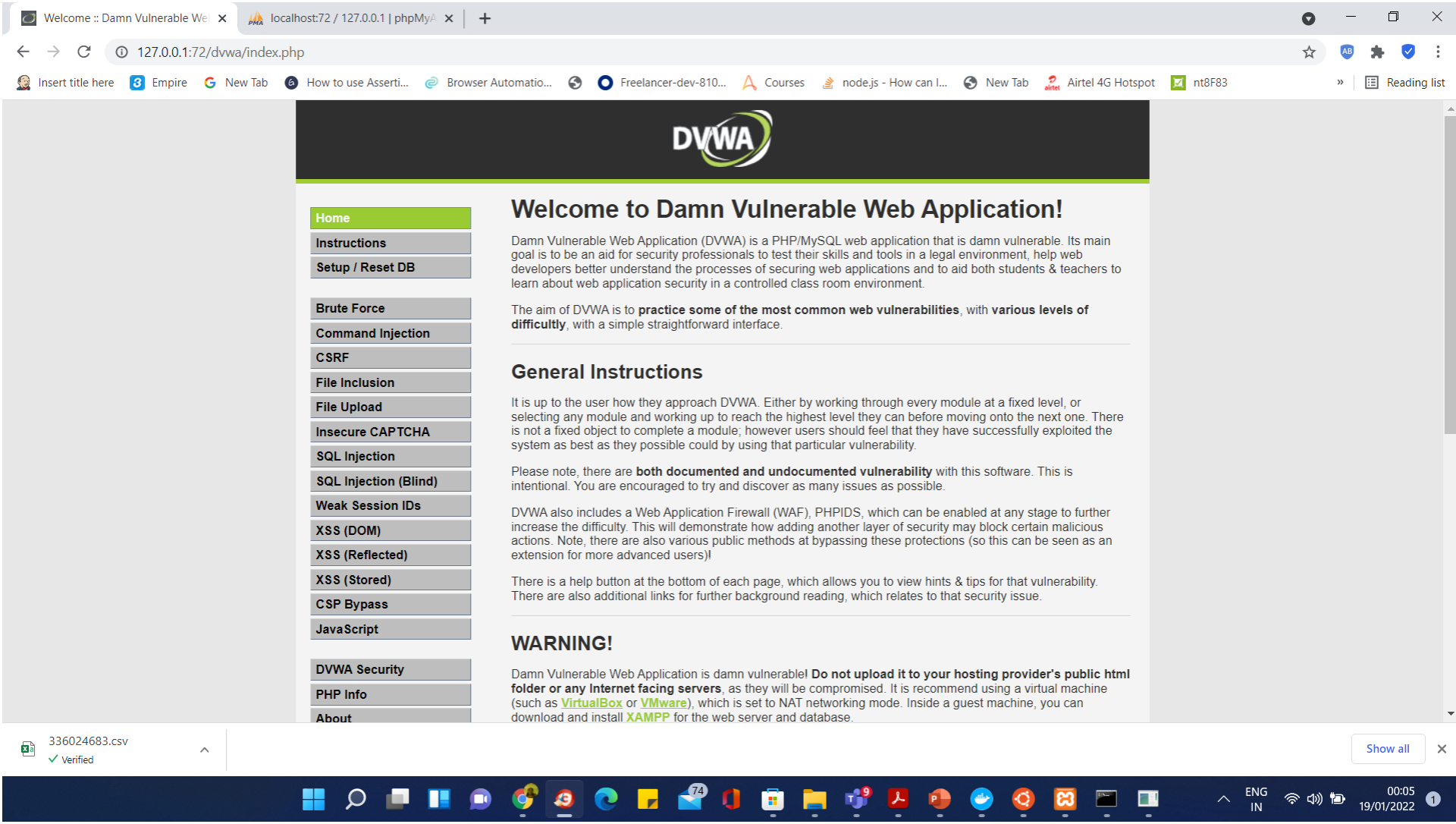
ZED Attack Proxy ZAP



Web Hacking DVWA



Web Hacking DVWA



OWASP Top 10 Vulnerabilities 2021

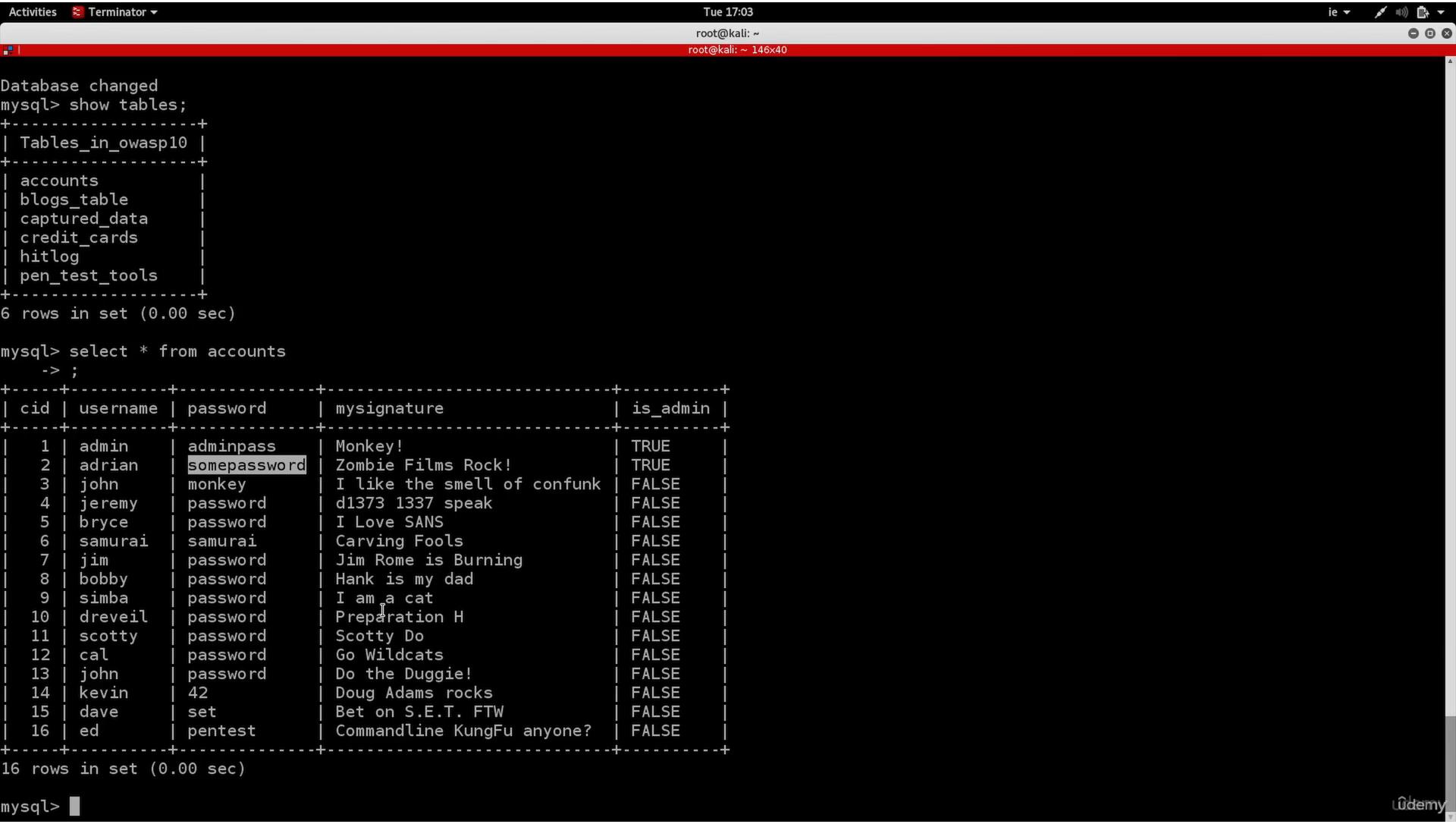
- Injection
- Broken Authentication
- Sensitive Data Exposure
- XML External Entities (XXE)
- Broken Access Control
- Security Misconfigurations
- Cross-Site Scripting (XSS)
- Insecure Deserialization
- Using Components with Known Vulnerabilities
- Insufficient Logging and Monitoring

SQL Injection

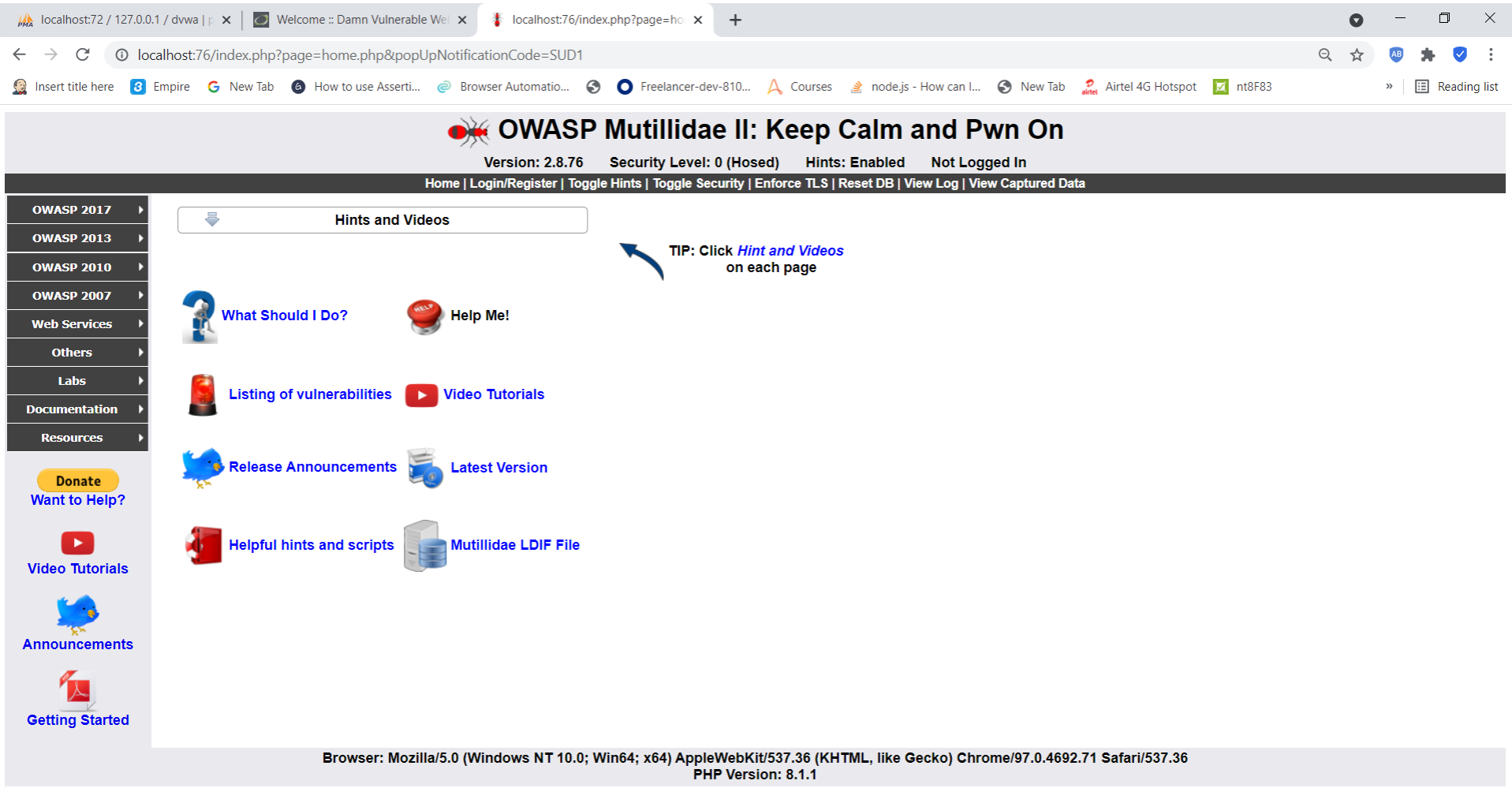
Why are they so dangerous

1. They are everywhere.
2. Give access to the database → sensitive data.
3. Can be used to read local files outside www root.
4. Can be used to log in as admin and further exploit the system.
5. Can be used to upload files.

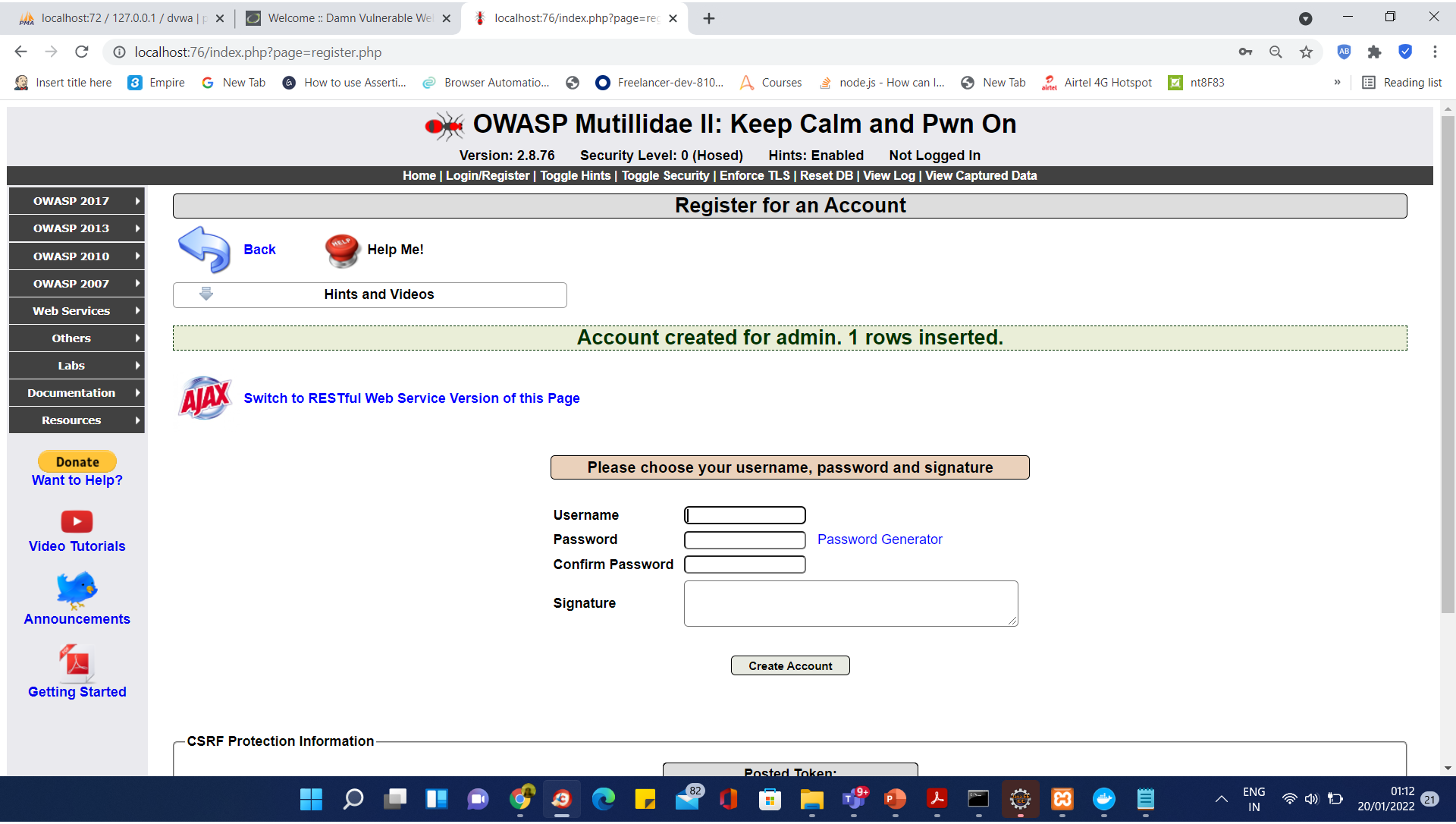
SQL Injection



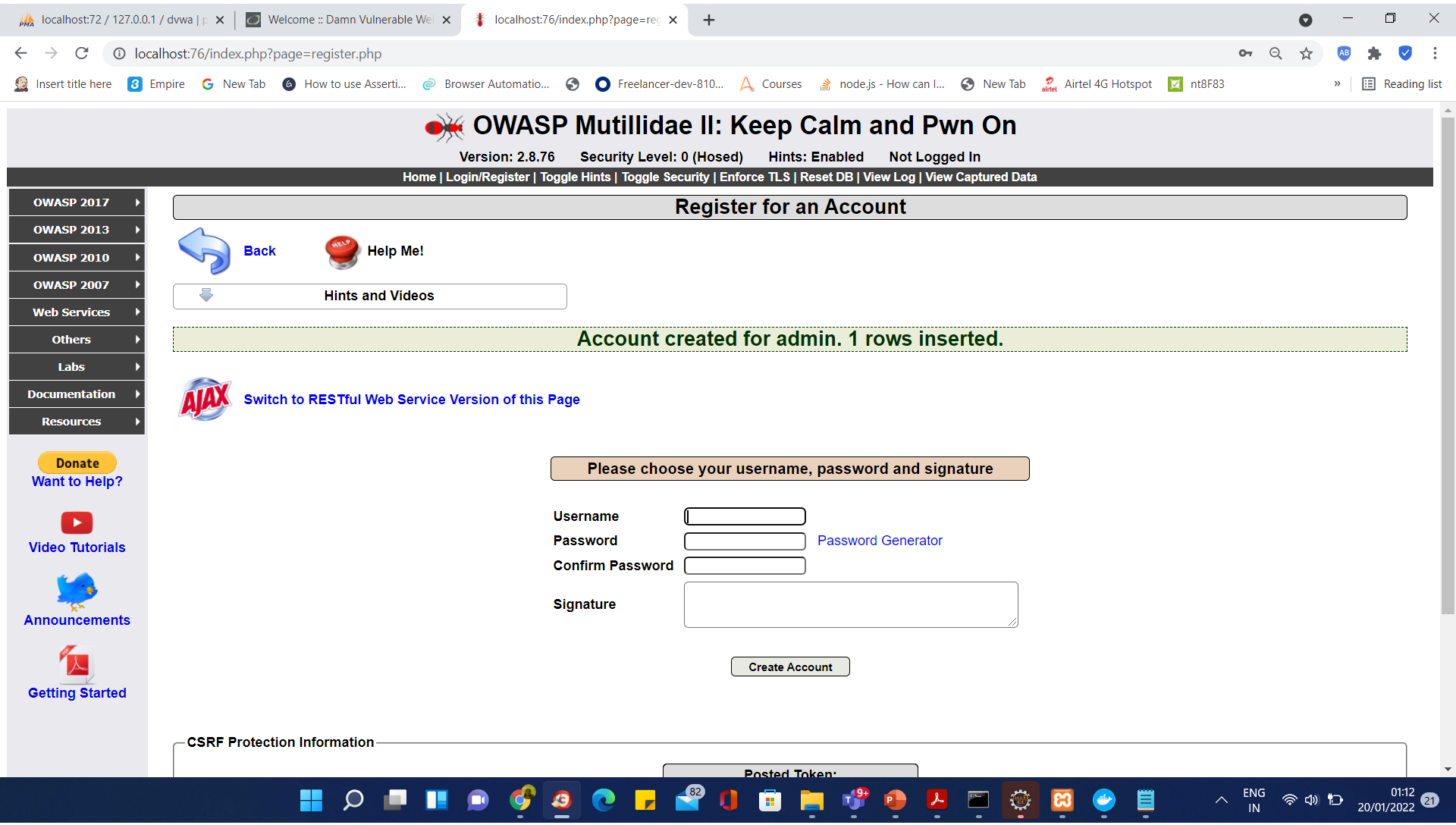
Mutillidae Project (dockerhub –mutillidae)



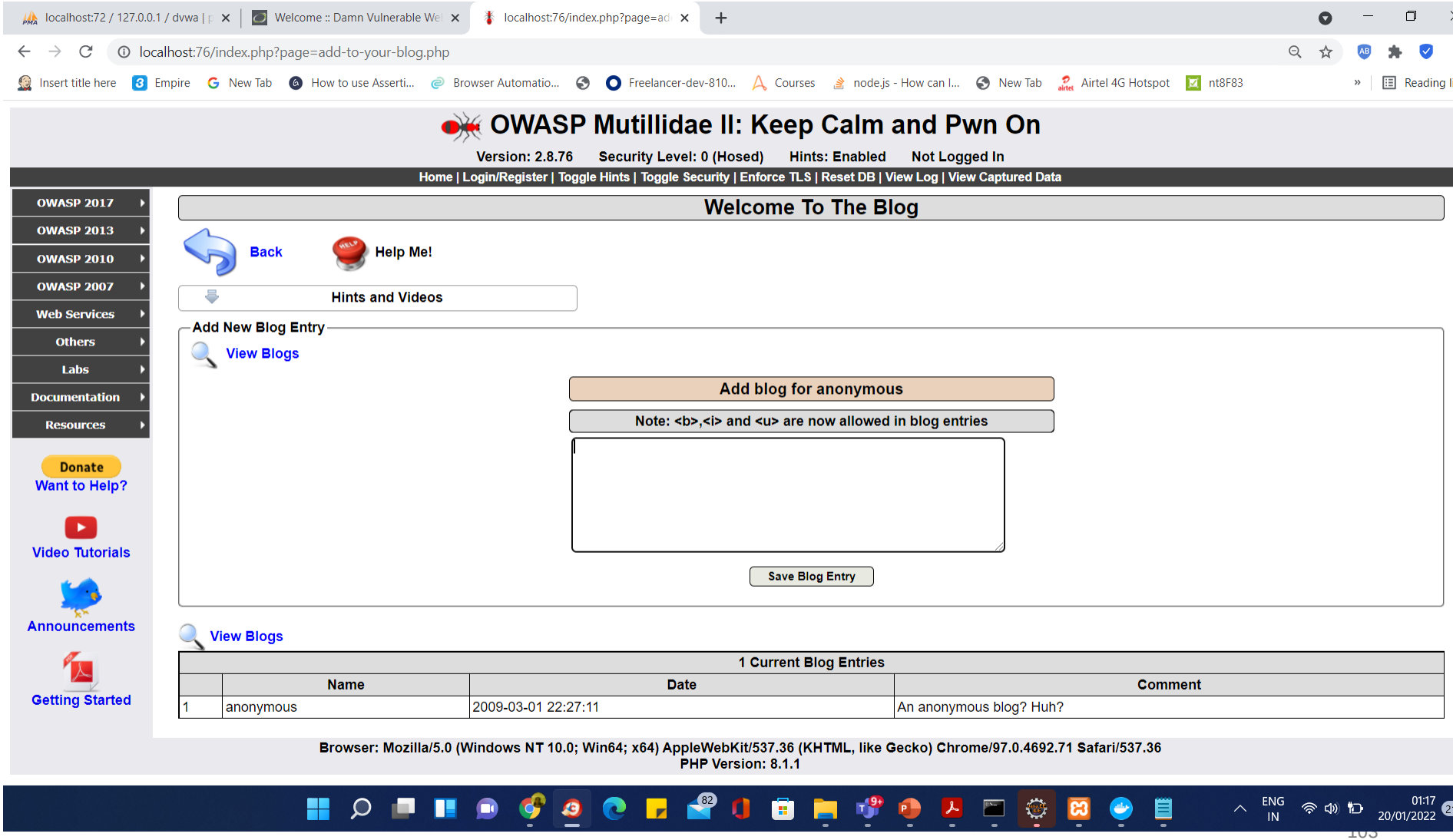
Sql Injection



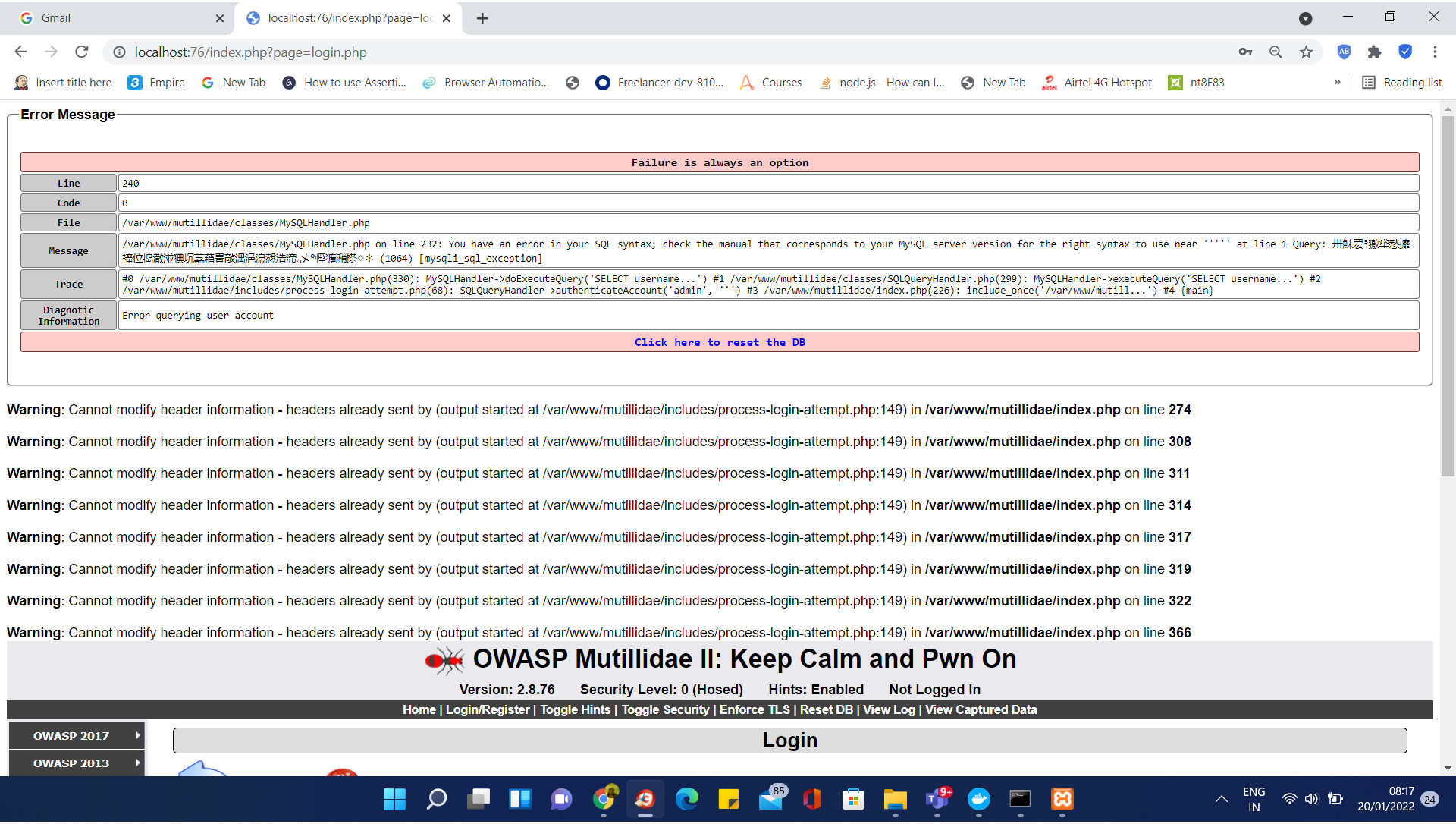
Sql Injection



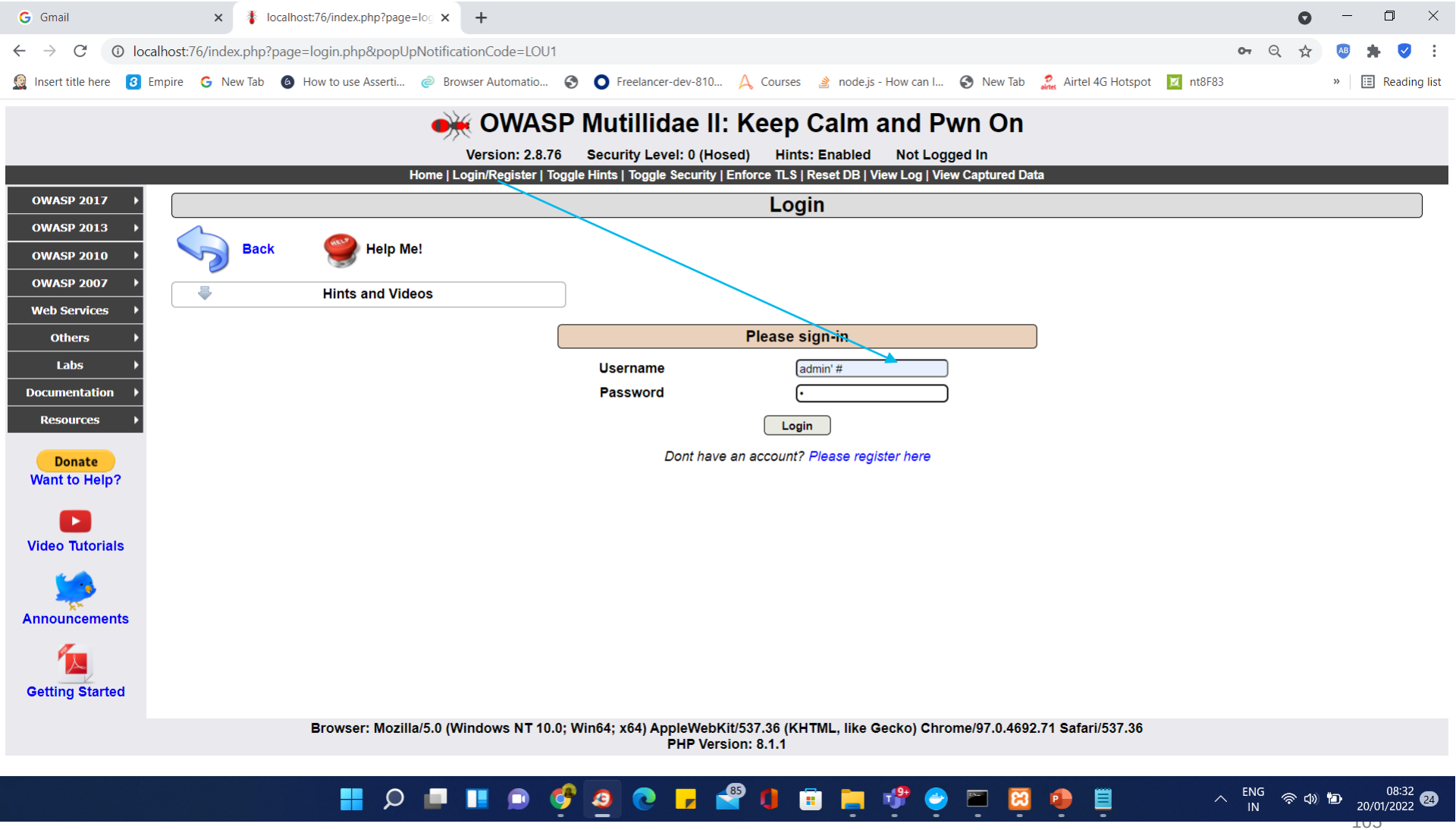
Sql Injection



Sql Injection



Sql Injection



Cross site Injection

Exploitation – XSS Vulns

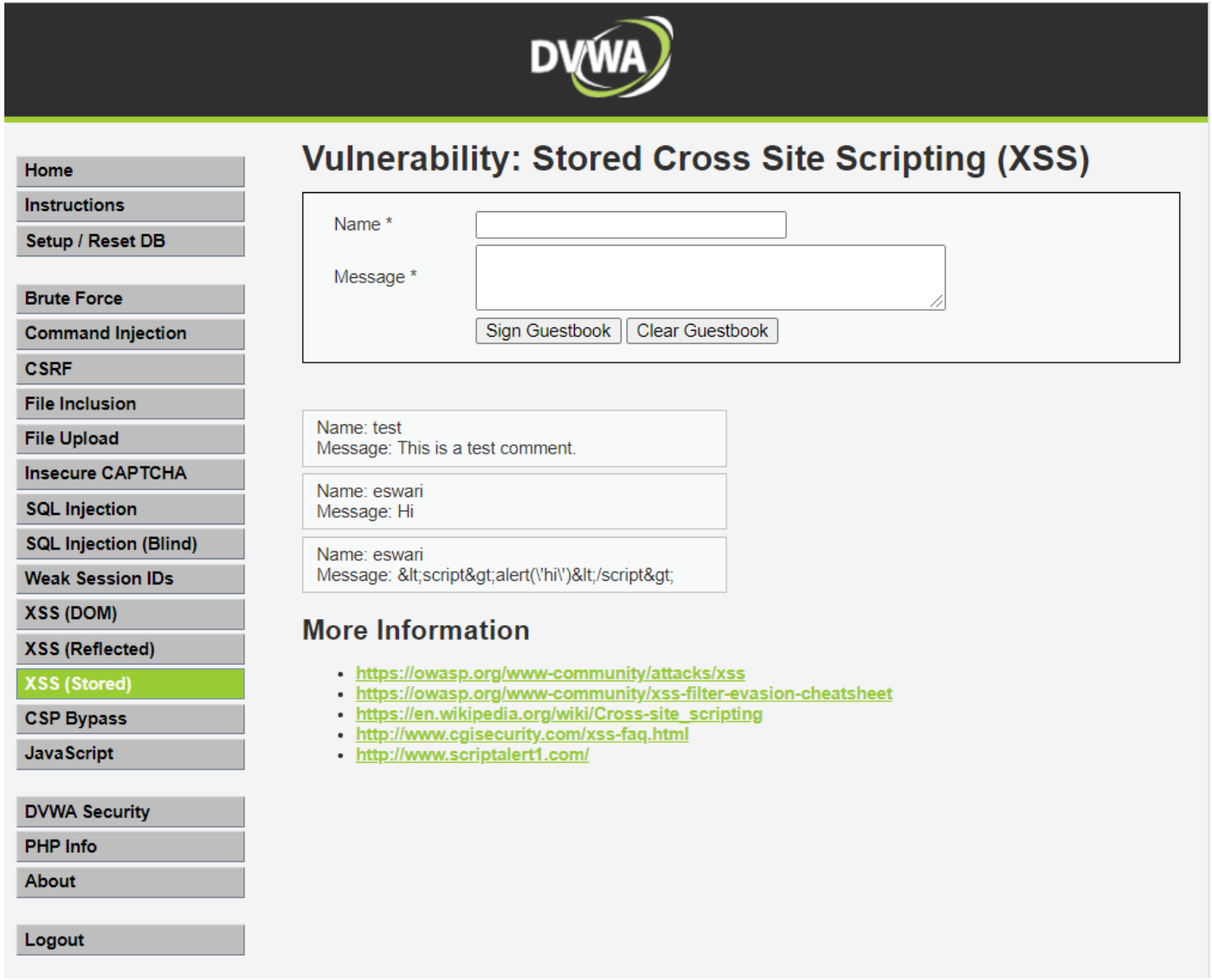
XSS – Cross Site Scripting vulns

- Allow an attacker to inject javascript code into the page.
- Code is executed when the page loads.
- Code is executed on the client machine not the server.

Three main types:

1. Persistent/Stored XSS
2. Reflected XSS
3. DOM based XSS

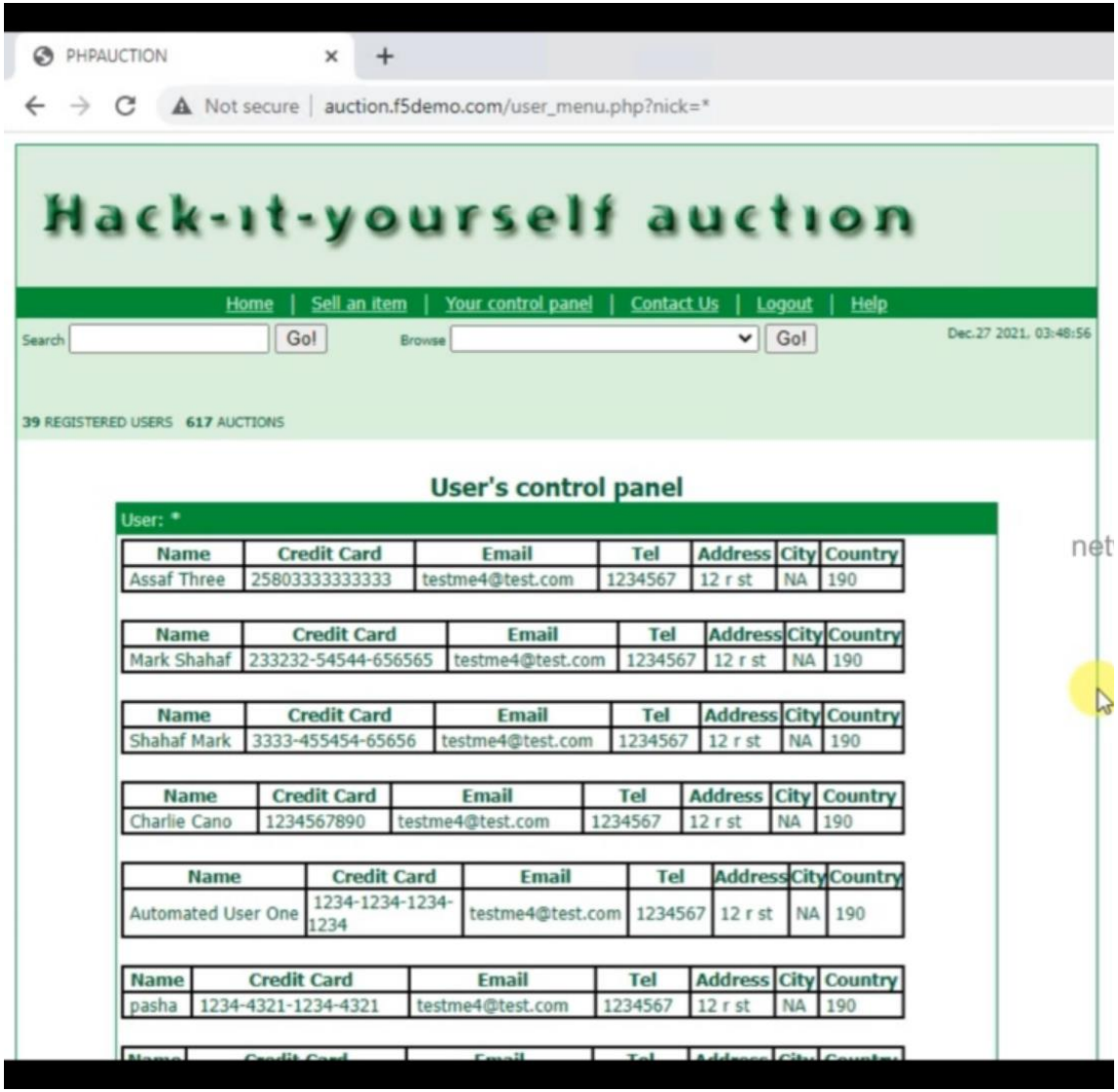
Cross site Injection



Parameter Tampering

- It is a technique of manipulating the parameter in the URL string to retrieve the data to which he is not authorized.
- Parameter Tampering is about changing the parameter in the request and also in the entity body in case of POST method
- This attack can be performed by a malicious user who wants to exploit the application for their own benefit, or an attacker who wishes to attack a third-person using a Man-in-the-middle attack. In both cases, tools likes Webscarab and Paros proxy are mostly used.

Parameter Tampering

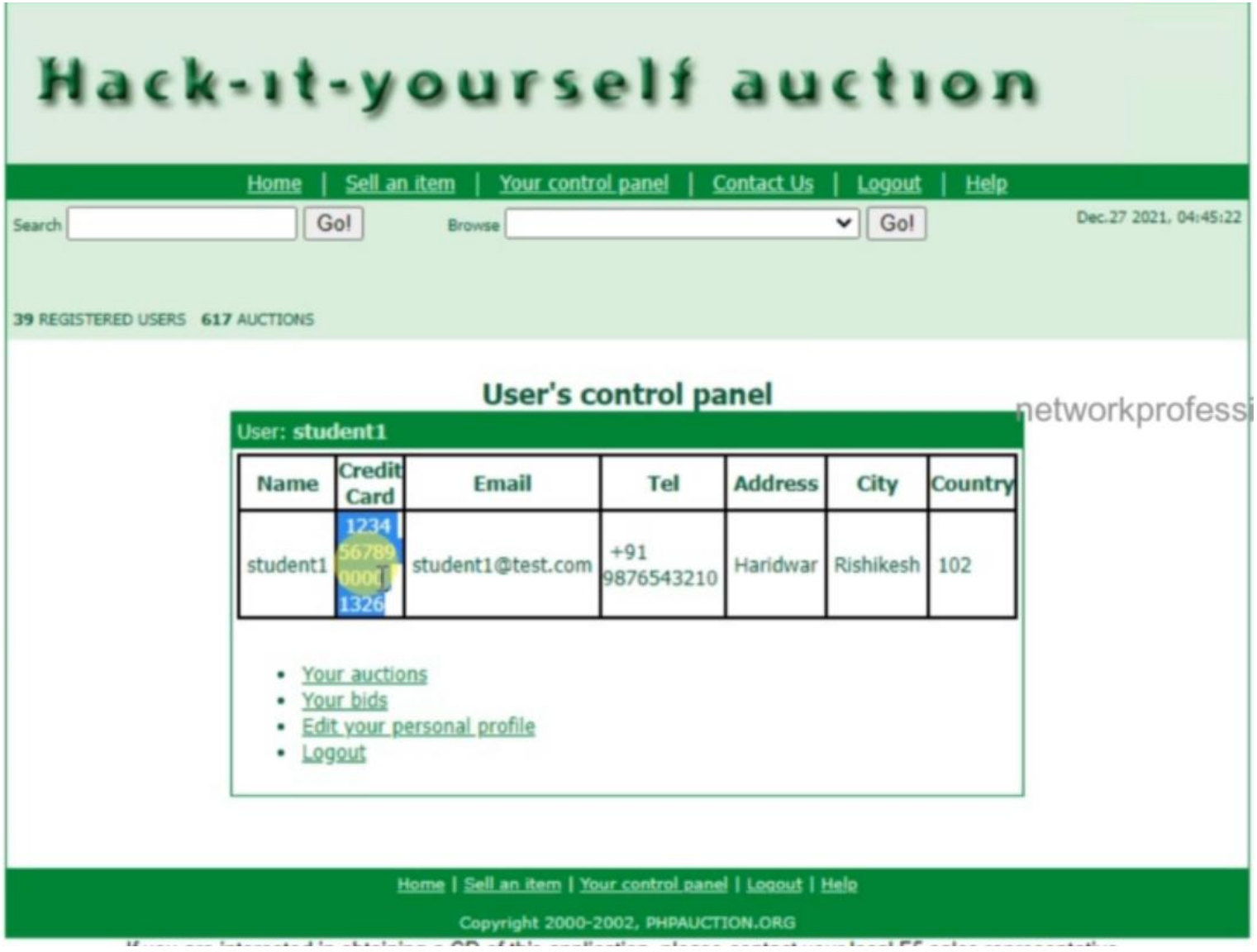


Sensitive Data Attack

- This security threat occurs when the web application doesn't adequately protect sensitive information like Credit card number, passwords, banking information, Social security number, location, health data, or any other similar crucial data whose leak can be critical for the user.

- This threat affects users the most and can cause financial loss, access to the victim's accounts, blackmailing which ultimately results in decreased trust in the brand.

Sensitive Data Attack



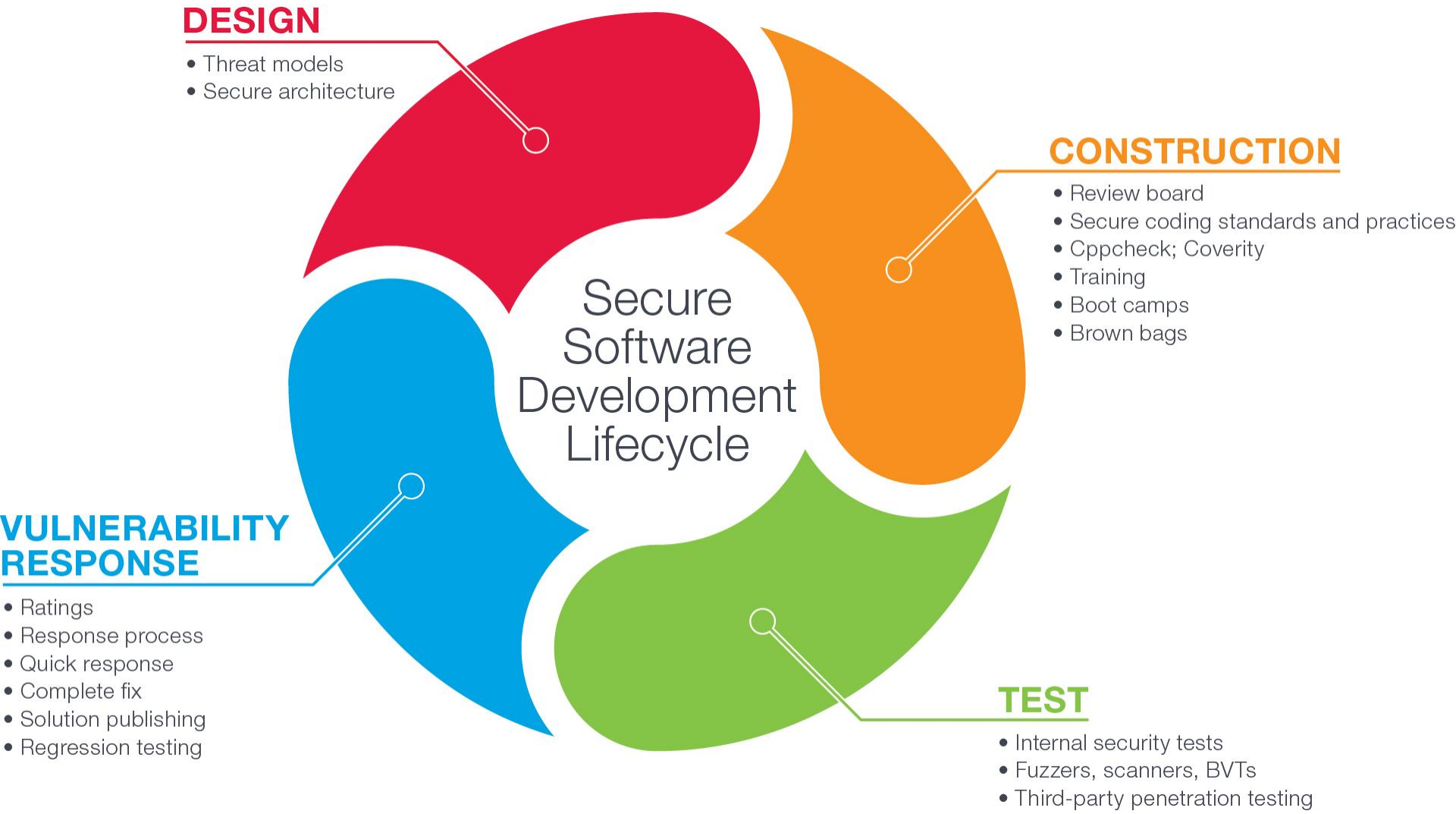
Forceful Browsing

- It refers to directly accessing a web page that should not be shown to just any other user or which he is not authorized.
- Web application that was not properly configured may allow the malicious user to directly access the URL that could contain sensitive information.
- Forceful browsing is editing the URL in browser to gain access to files that the owner of the site didn't intend to be publicly accessible.

Penetration Test Report

- Sample+Pentest+Report.docx

Secure Software Development



Need for Secure Coding

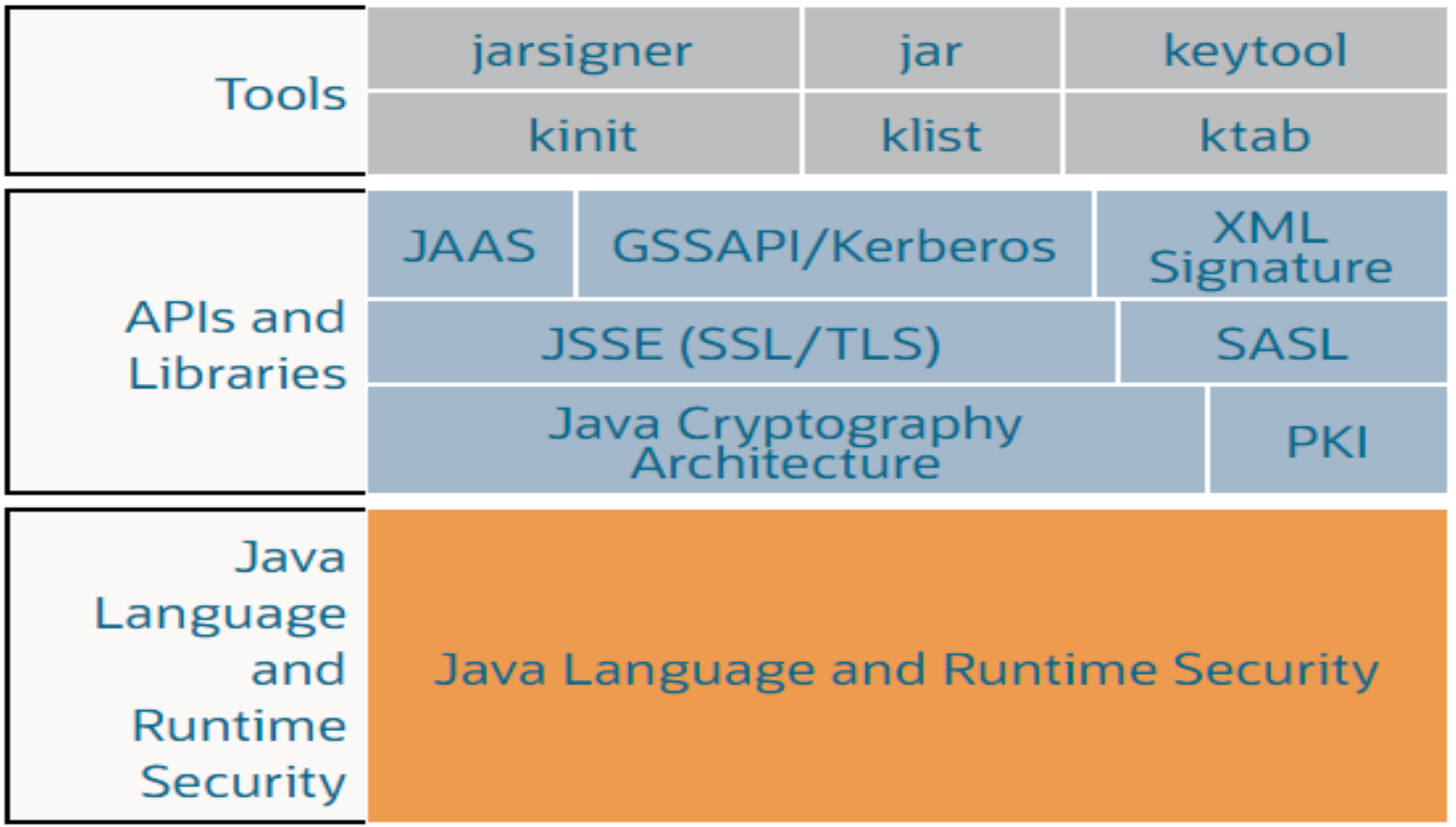
- Secure coding is the practice of writing software that's protected from vulnerabilities.

Java SE Security Mechanisms

- Java Authentication and Authorization Service (JAAS)
- Java Generic Security Services (Java GSS-API)
- Java Cryptography Extension (JCE)
- Java Secure Sockets Extension (JSSE)
- Simple Authentication and Security Layer (SASL)

Java SE Security Mechanisms

Java security's components are structured as follows:



Digitally sign JARs with jarsigner

- When distributing applications as JAR, EAR or WAR files, it's a good practice if other users are downloading archives over the public internet, to digitally sign JAR files with jarsigner.
- The jarsigner tool is bundled with every JDK installation, is found in the JDK's bin directory.
- It is likely accessibly directly through a command prompt or terminal window so long as the JDK's bin directory has been put on your operating system's PATH.

Digitally sign JARs with jarsigner

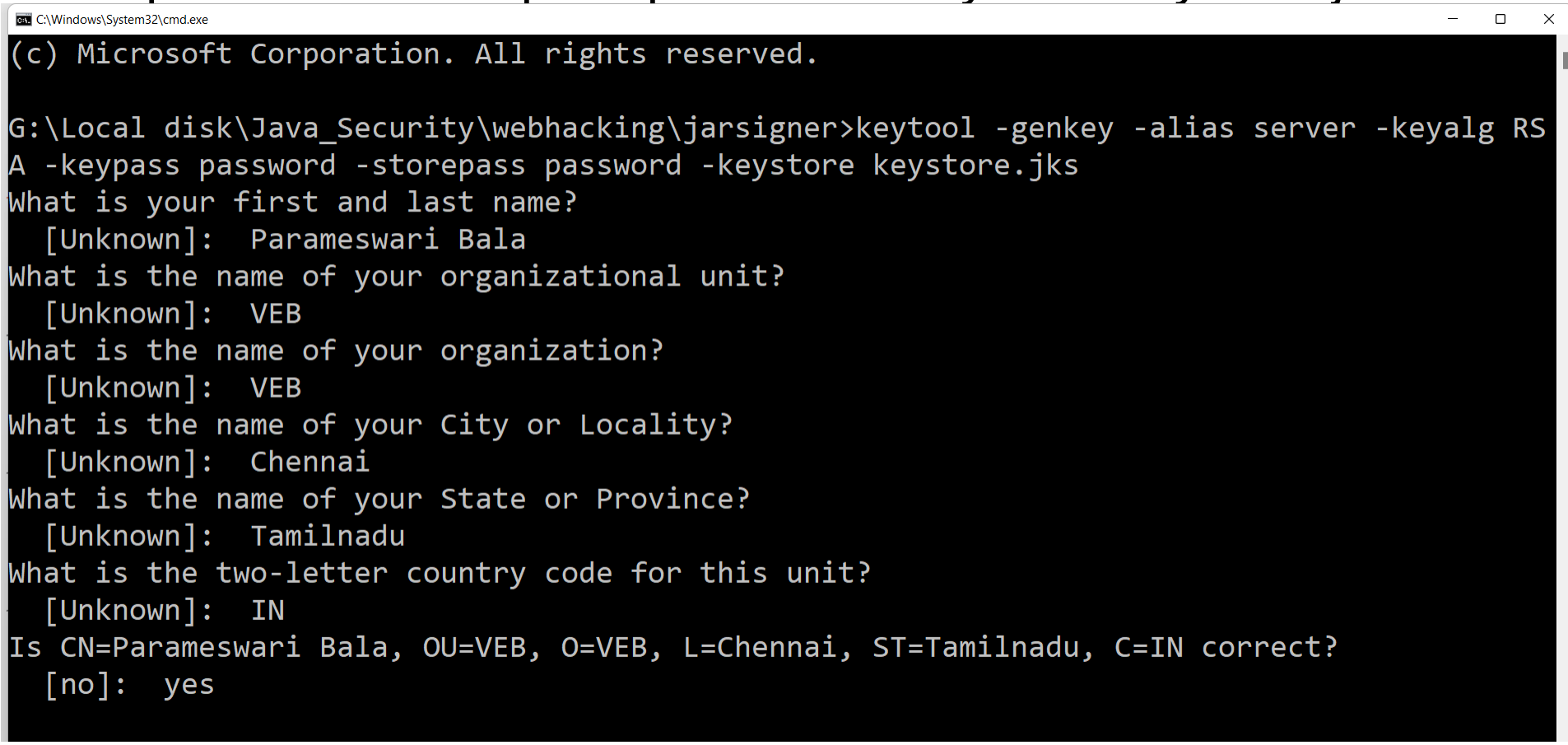
- To sign a JAR with jarsigner, you first need to create a public and private key.
- The private key will sign the JAR, and the public key will be able to attest to the veracity of the signature.
- The JDK’s keytool can be used to create the public and private keys and have them stored in a local keystore.

Digitally sign JARs with jarsigner

- Create a JAR file with Java's JAR utility
- Create public and private keys with Java's keytool
- Export the server-side digital certificate with the keytool
- Use the jarsigner tool to sign the JAR file digitally
- Use the jarsigner command's -verify switch to validate the JAR file signing operation

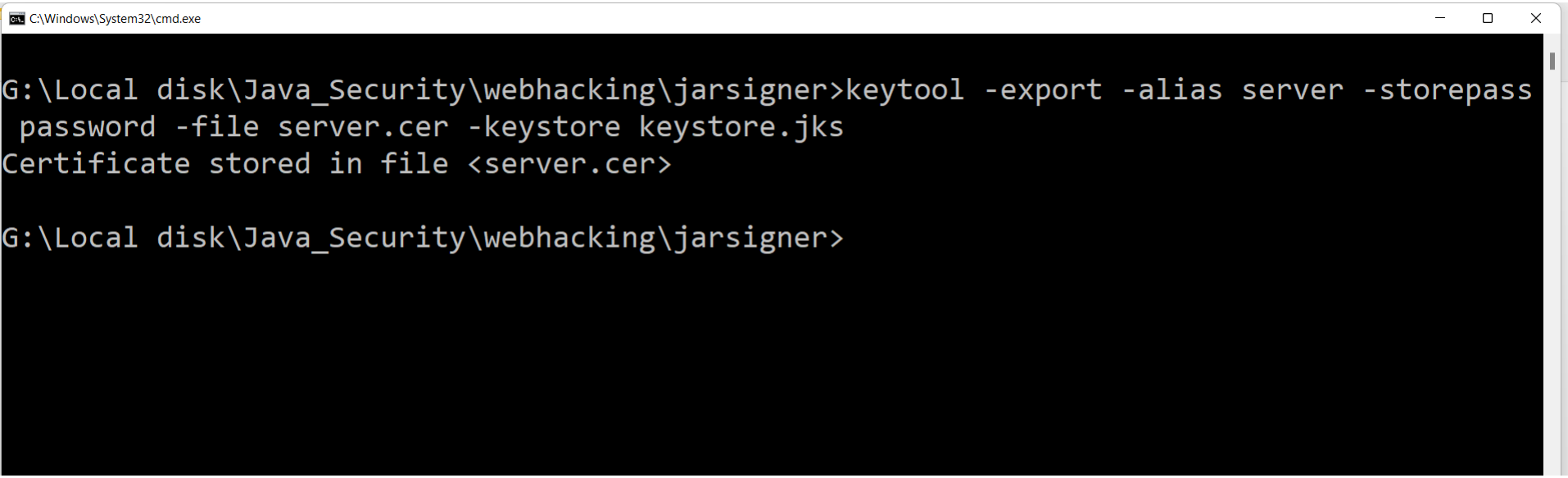
Digitally sign JARs with jarsigner

- Digital certificate generation with keytool
  - keytool -genkey -alias server -keyalg RSA -keypass password -storepass password -keystore keystore.jks



Java KeyTool Export

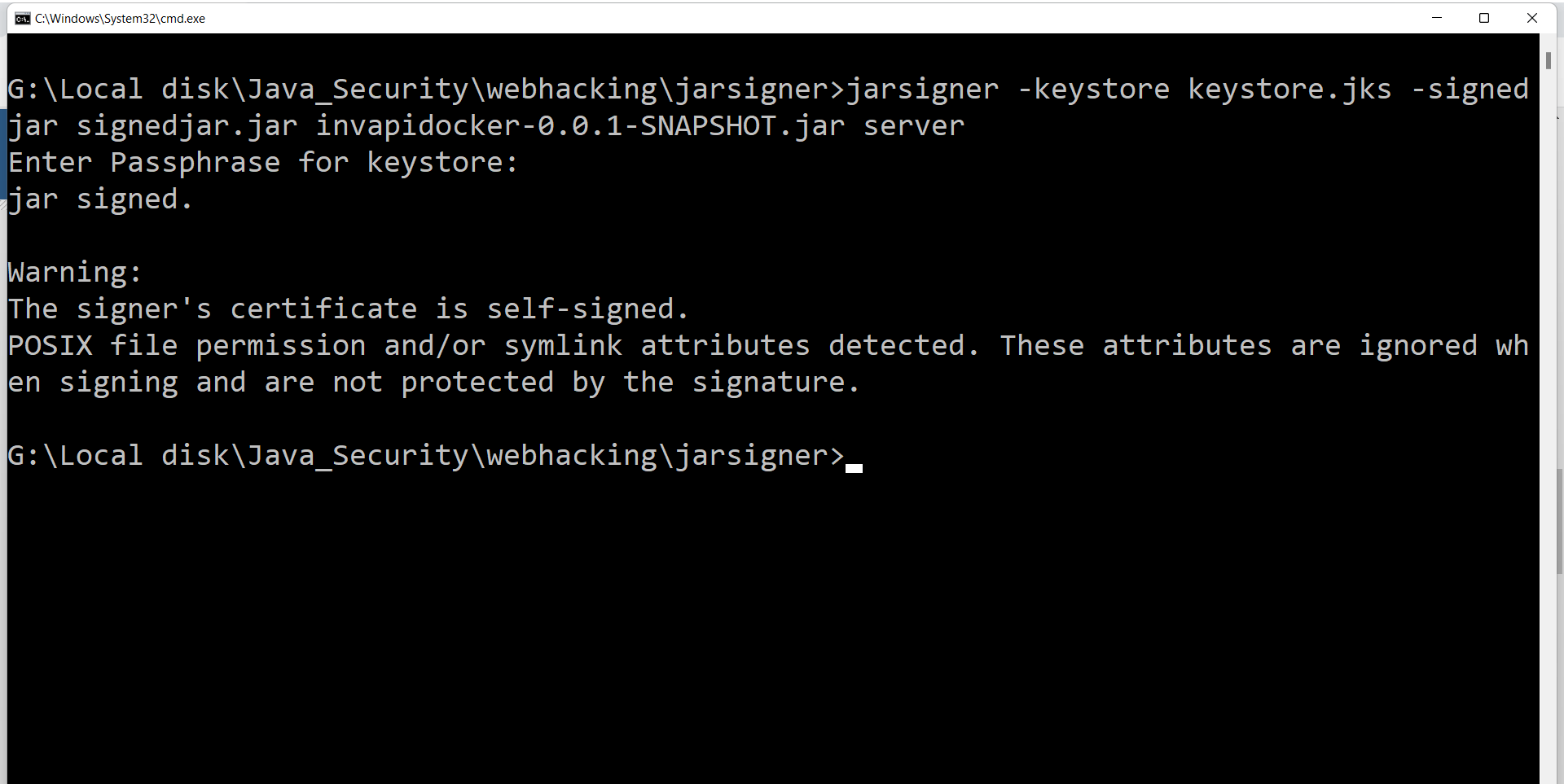
- keytool -export -alias server -storepass password -file server.cer -keystore keystore.jks



Password is password

Jar Signer

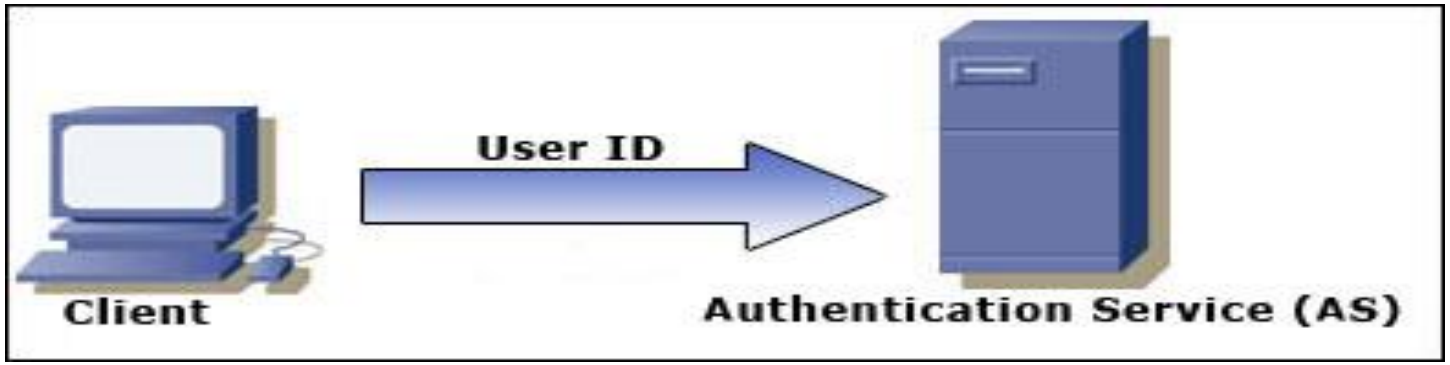
- jarsigner -keystore keystore.jks -signedjar signedjar.jar invapidocker-0.0.1-SNAPSHOT.jar server



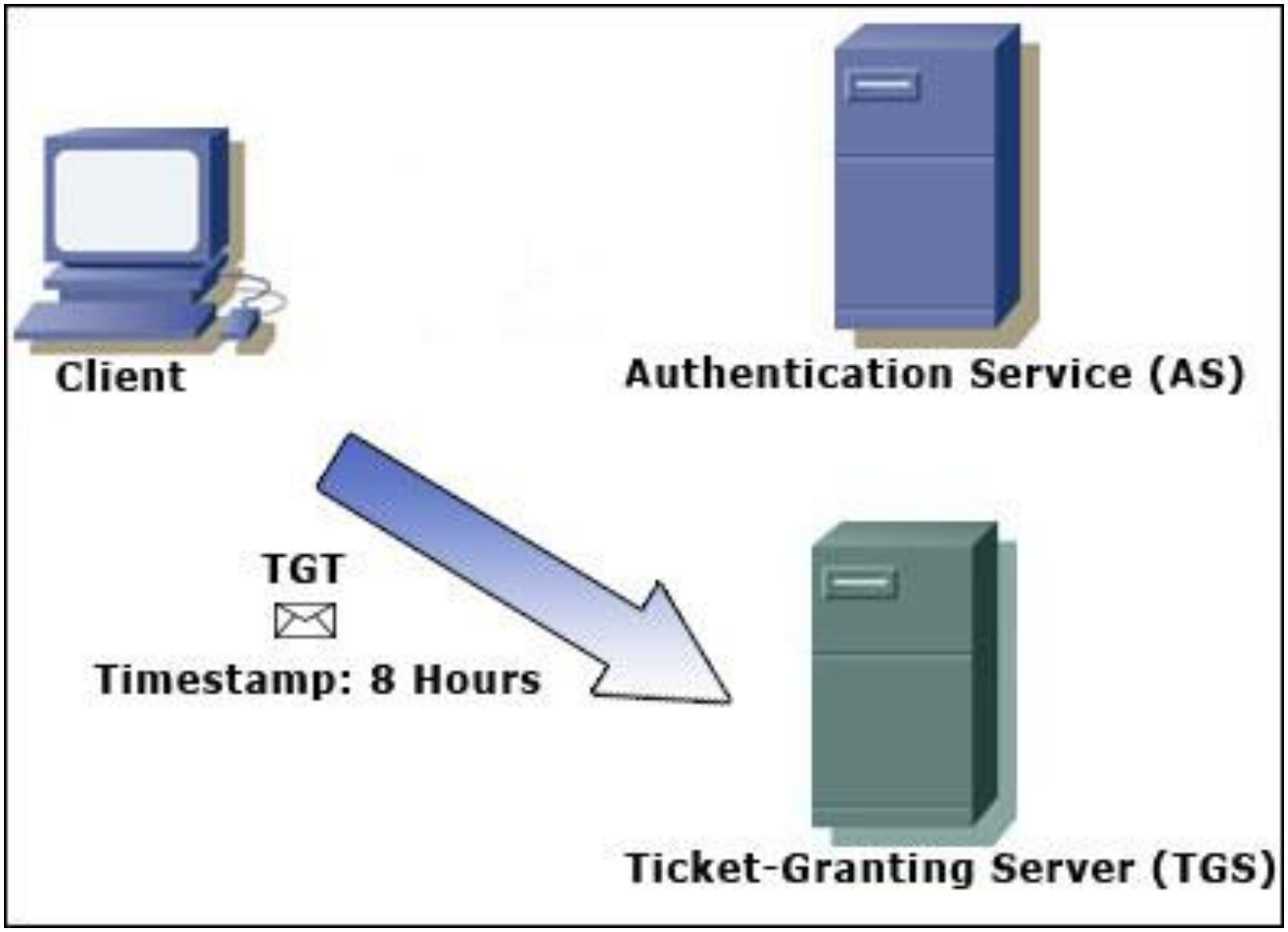
Verify a signed JAR file

- jarsigner -verify signedjar.jar

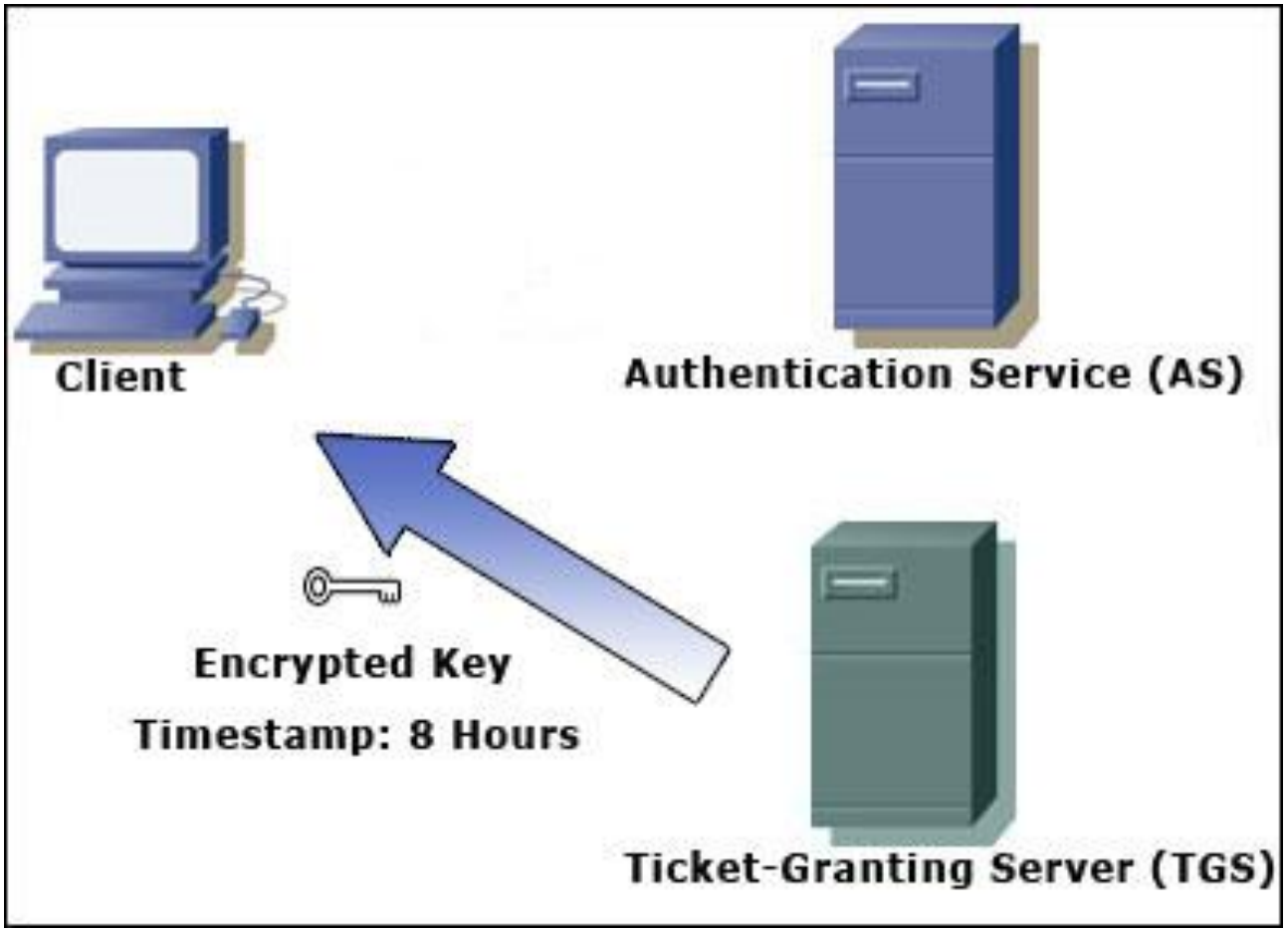
Kerberos (kinit,ktab etc.,)



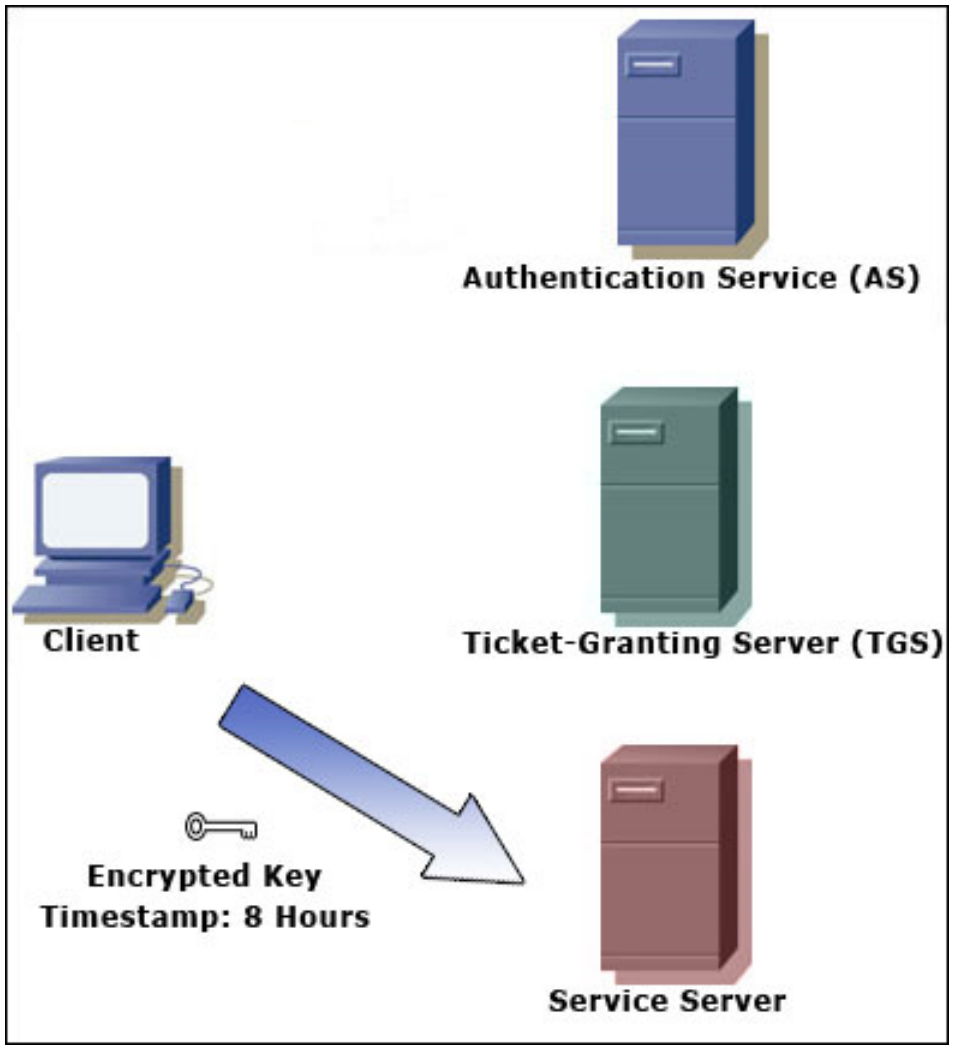
Kerberos


Client
Authentication Service (AS)
TGT
Timestamp: 8 Hours
Ticket-Granting Server (TGS)

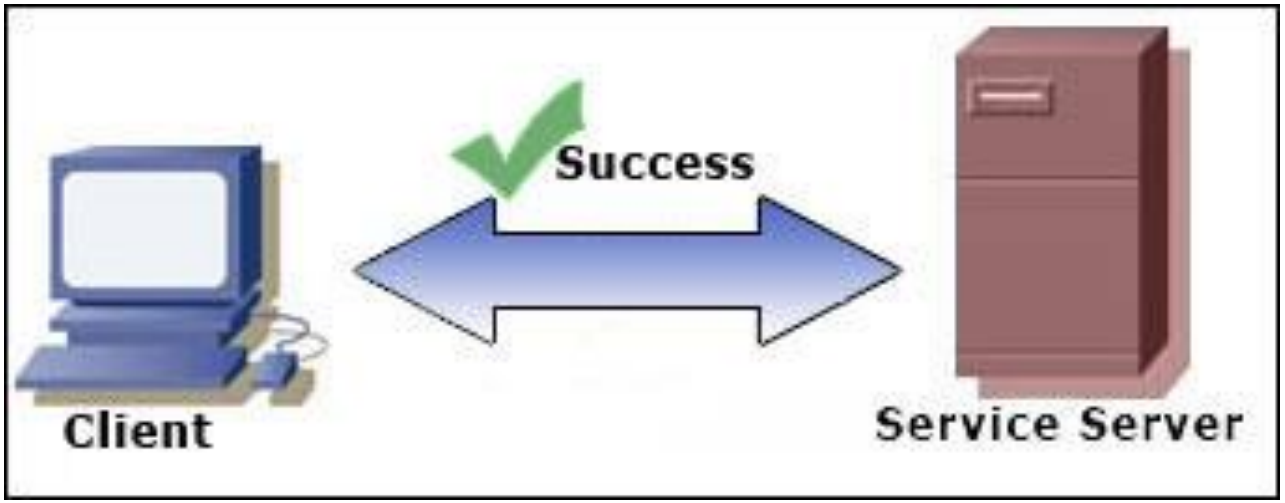
Kerberos


Client
Authentication Service (AS)
Encrypted Key
Timestamp: 8 Hours
Ticket-Granting Server (TGS)

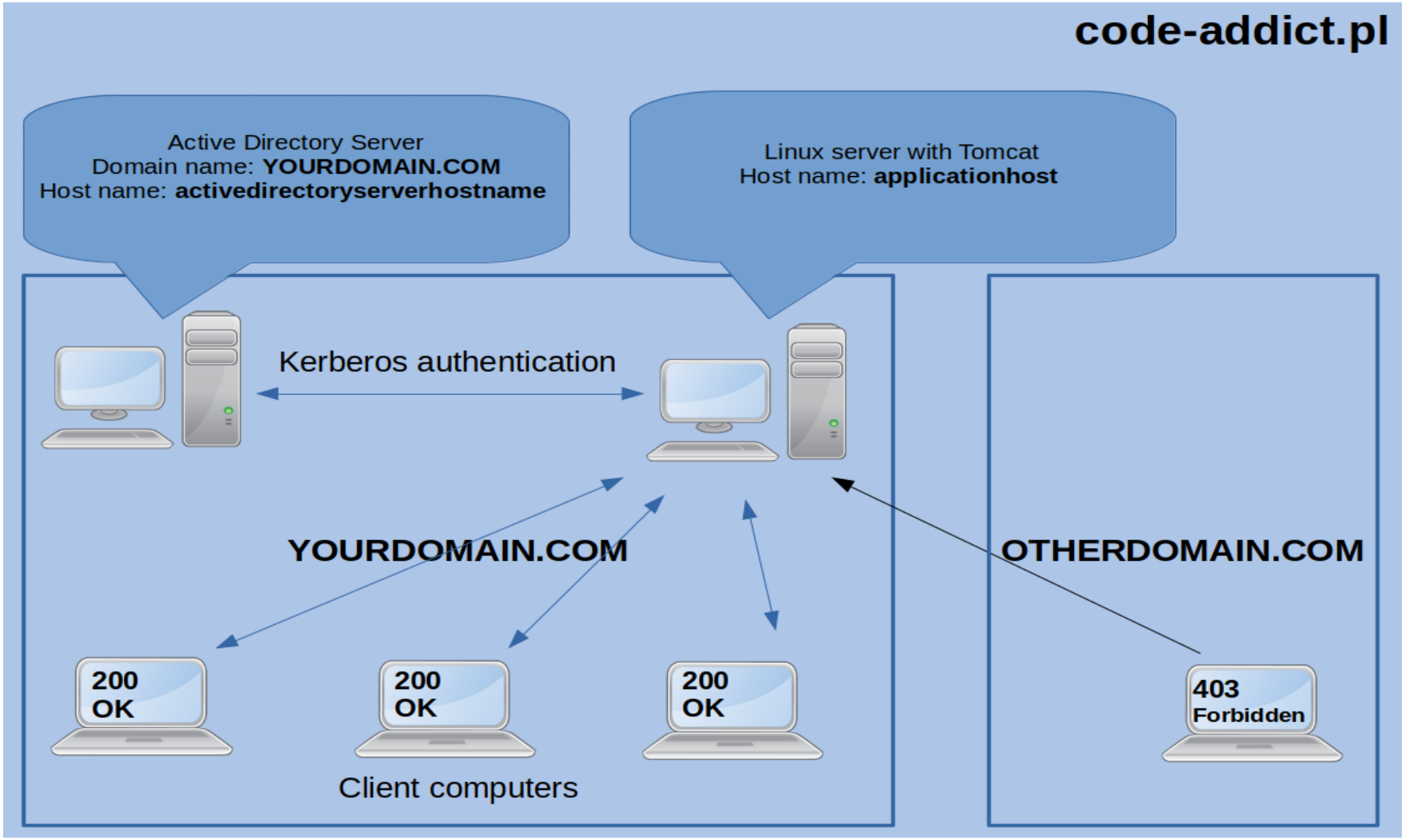
Kerberos


Authentication Service (AS)
Client
Ticket-Granting Server (TGS)
Encrypted Key
Timestamp: 8 Hours
Service Server

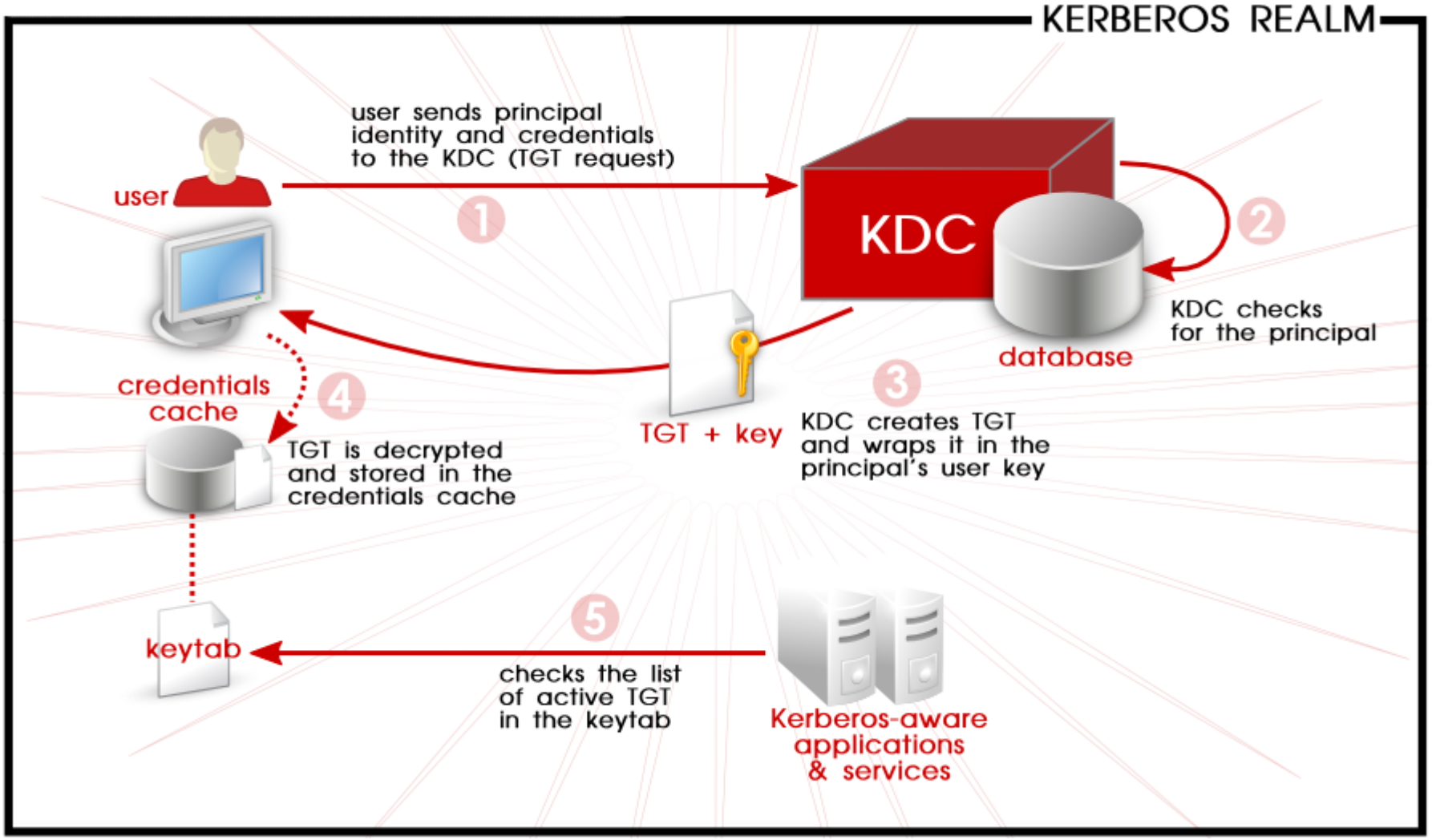
Kerberos



Kerberos


code-addict.pl
Active Directory Server
Domain name: YOURDOMAIN.COM
Host name: activedirectoryserverhostname
Linux server with Tomcat
Host name: applicationhost
Kerberos authentication
YOURDOMAIN.COM
OTHERDOMAIN.COM
200
OK
200
OK
200
OK
403
Forbidden
Client computers

Kerberos


KERBEROS REALM
user sends principal identity and credentials to the KDC (TGT request)
1
user
KDC
2
KDC checks for the principal
database
TGT + key
3
KDC creates TGT and wraps it in the principal's user key
credentials cache
4
TGT is decrypted and stored in the credentials cache
keytab
5
checks the list of active TGT in the keytab
Kerberos-aware applications & services

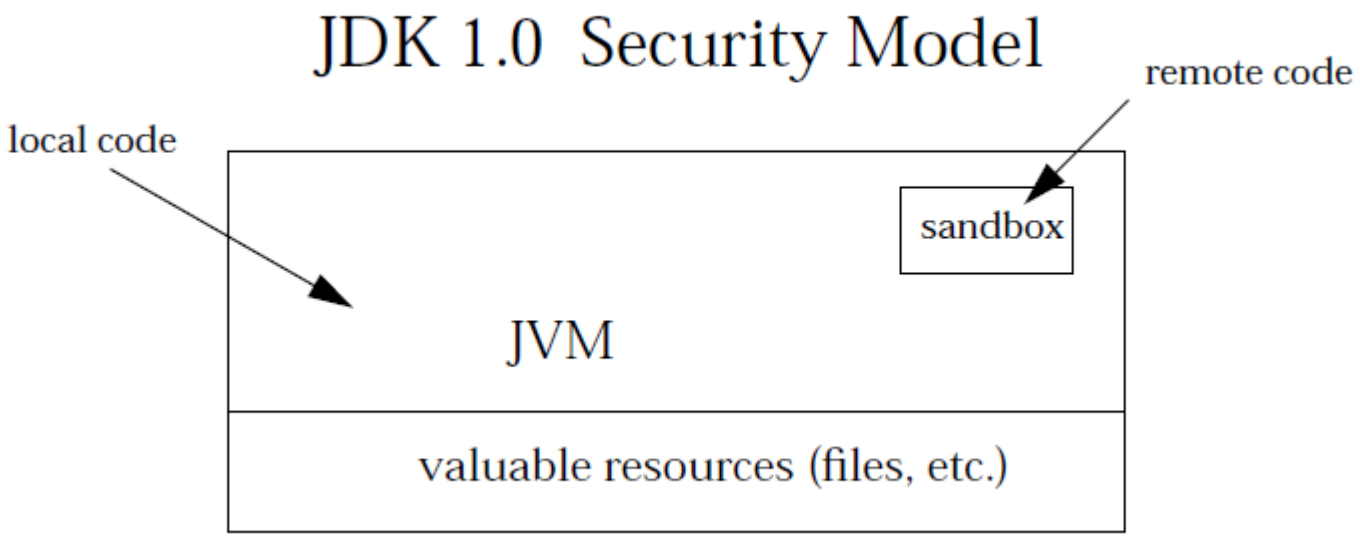
Java Language and runtime security

- Java security includes two aspects:
  - Provide the Java platform as a secure, ready-built platform on which to run Java-enabled applications in a secure fashion.
  - Provide security tools and services implemented in the Java programming language that enable a wider range of security-sensitive applications, for example, in the enterprise world.

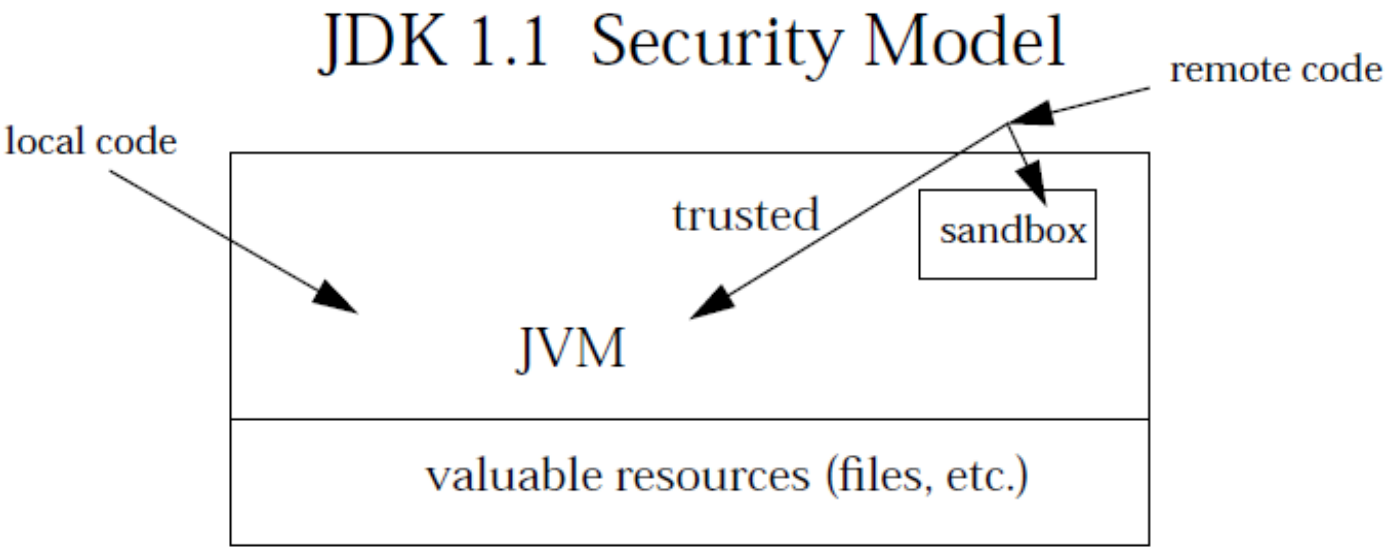
The Original Sandbox Model

- The original security model provided by the Java platform is known as the sandbox model, which existed in order to provide a very restricted environment in which to run untrusted code obtained from the open network.
- The essence of the sandbox model is that local code is trusted to have full access to vital system resources (such as the file system) while downloaded remote code (an applet) is not trusted and can access only the limited resources provided inside the sandbox

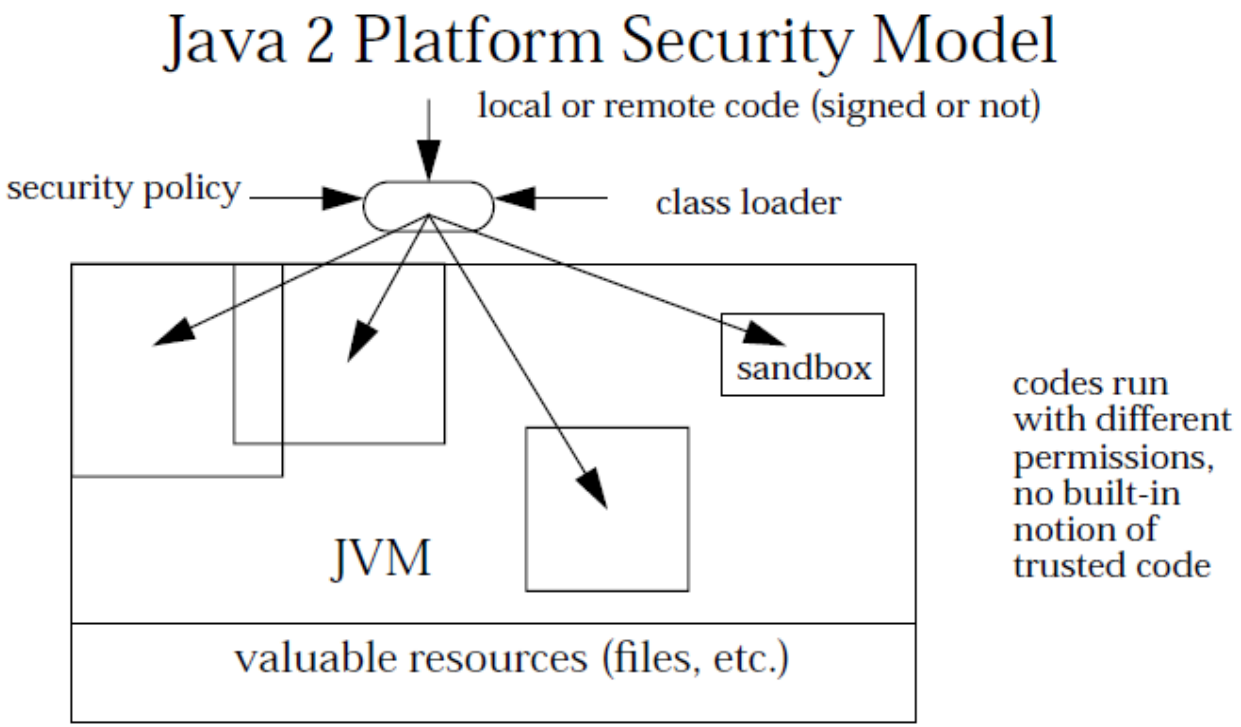
The Original Sandbox Model



The Original Sandbox Model

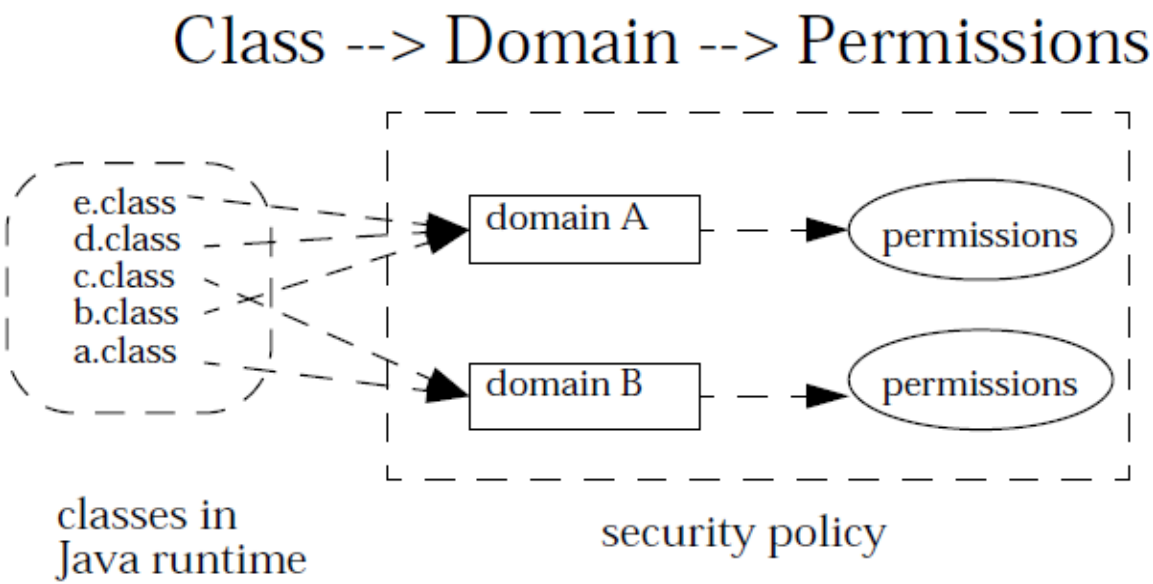


The Original Sandbox Model

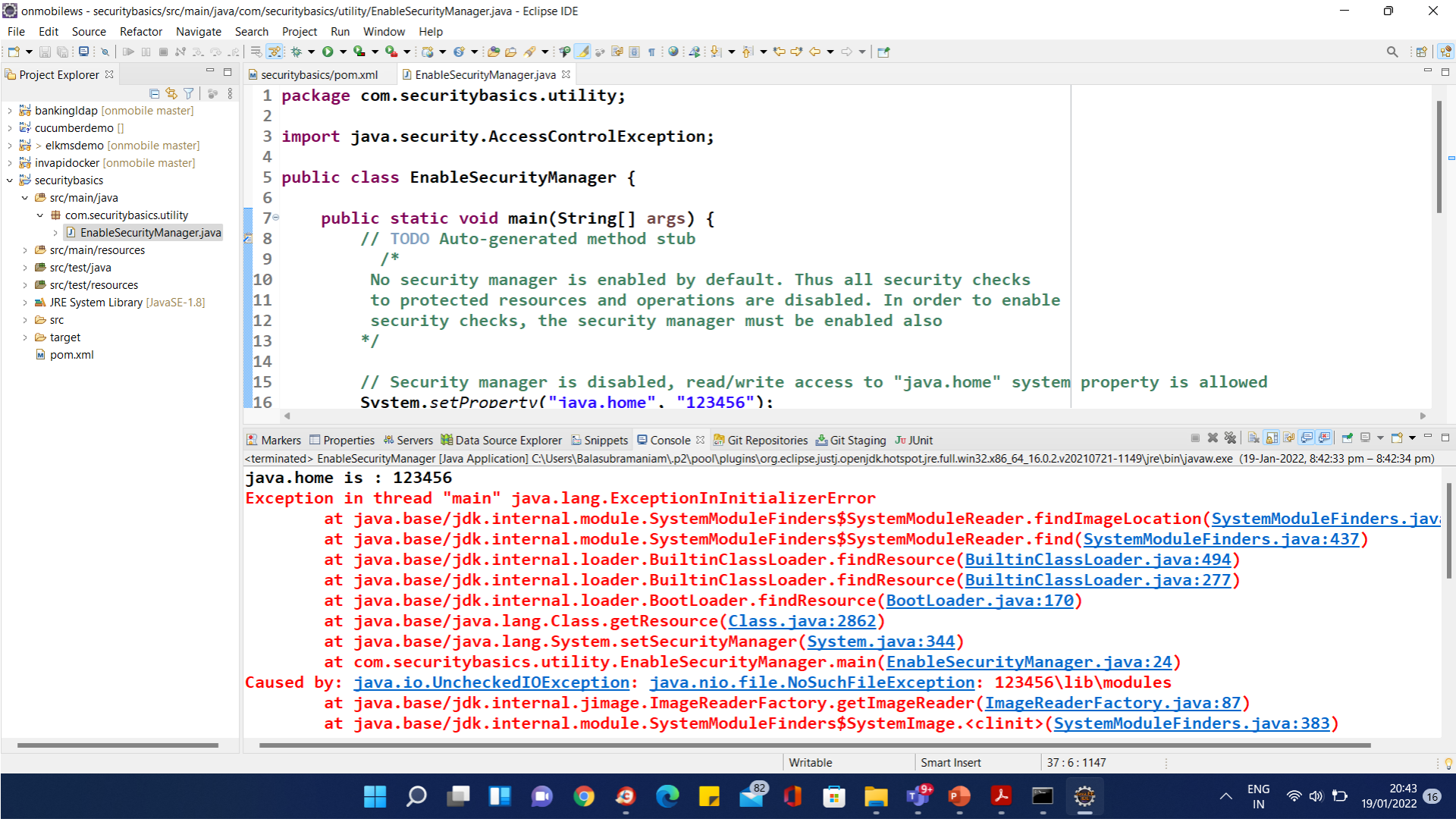


Code runs with different permissions, no built-in notion of trusted codes

New Protection Mechanisms



New Protection Mechanisms



Secure Coding in Java

- https://www.oracle.com/java/technologies/javase/seccodeguide.html

Secure Coding in Java

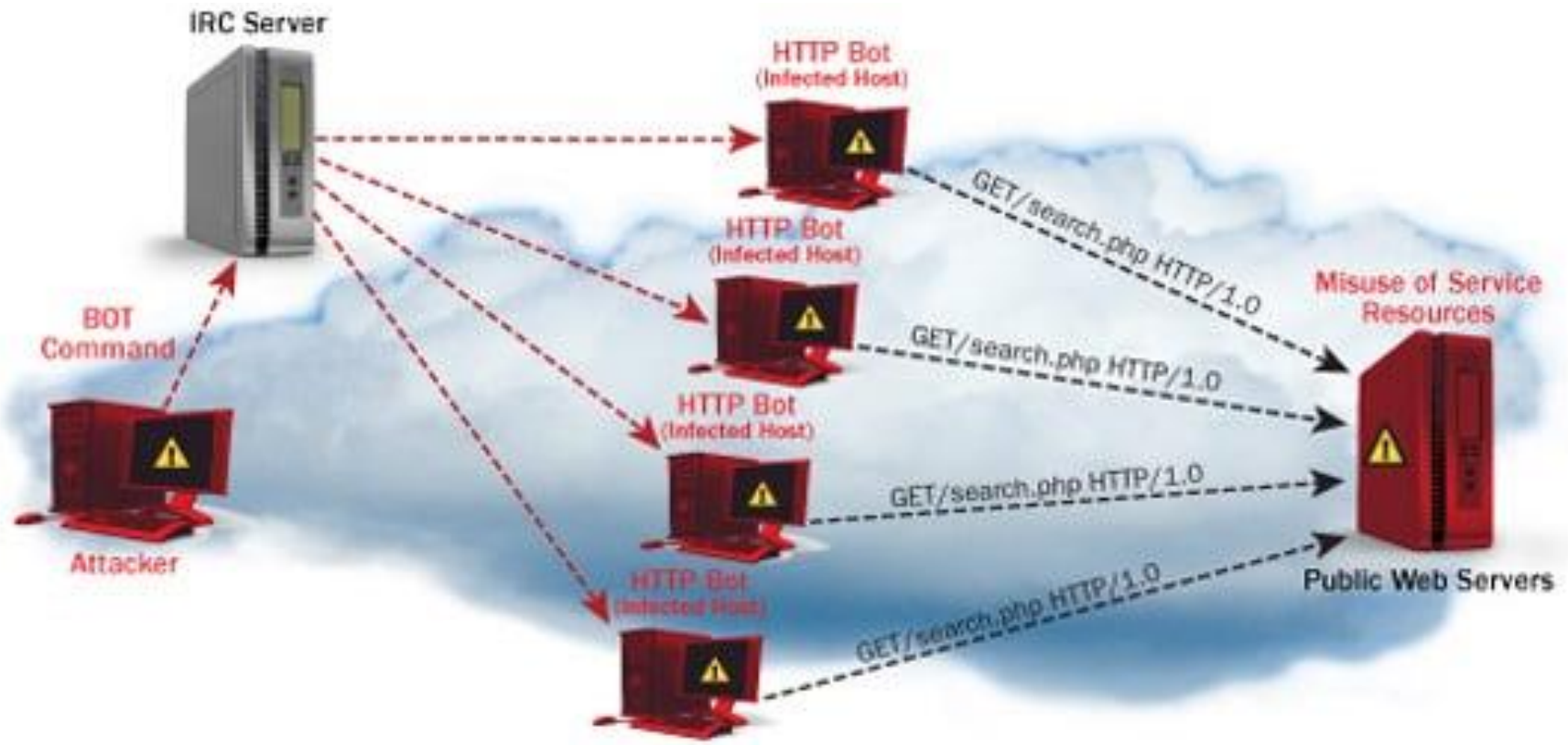
- rule #1: Write clean, strong Java code
- rule #2: Avoid serialization
- rule #3: Never expose unencrypted credentials or PII
- rule #4: Use known and tested libraries
- rule #5: Be paranoid about external input
- **Use frameworks like Hibernate and Spring Data JPA for the data layer of an application.**
- rule #6: Always use prepared statements to handle SQL parameters (only for jdbc)
- rule #7: Don't reveal implementation via error messages

Secure Coding in Java

- rule #8: Keep security releases up to date
- rule #9: Look for dependency vulnerabilities
- rule #10: Monitor and log user activity
- rule #11: Watch out for Denial of Service (DoS) attacks
- rule #12: Consider using the Java security manager
- rule #13: Consider using an external cloud authentication service

Find bugs Clean Code

Denial Of Service



DoS Attack Tools

- Nemesy– this tool can be used to generate random packets. It works on windows. This tool can be downloaded from http://packetstormsecurity.com/files/25599/nemesy13.zip.html . Due to the nature of the program, if you have an antivirus, it will most likely be detected as a virus.
- Land and LaTierra– this tool can be used for IP spoofing and opening TCP connections
- Blast– this tool can be downloaded from http://www.opencomm.co.uk/products/blast/features.php
- Panther– this tool can be used to flood a victim's network with UDP packets.
- Botnets– these are multitudes of compromised computers on the Internet that can be used to perform a distributed denial of service attack.

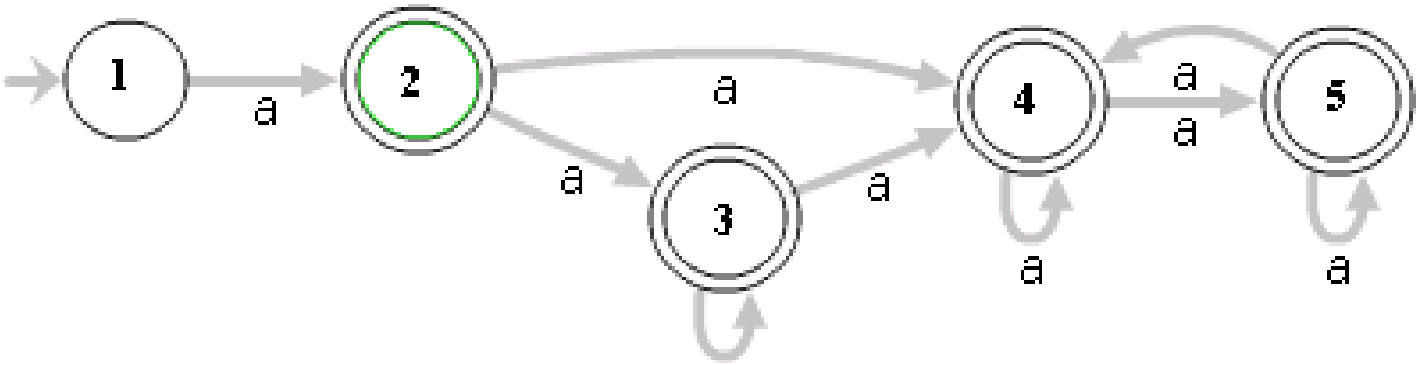
Secure Coding -DOS

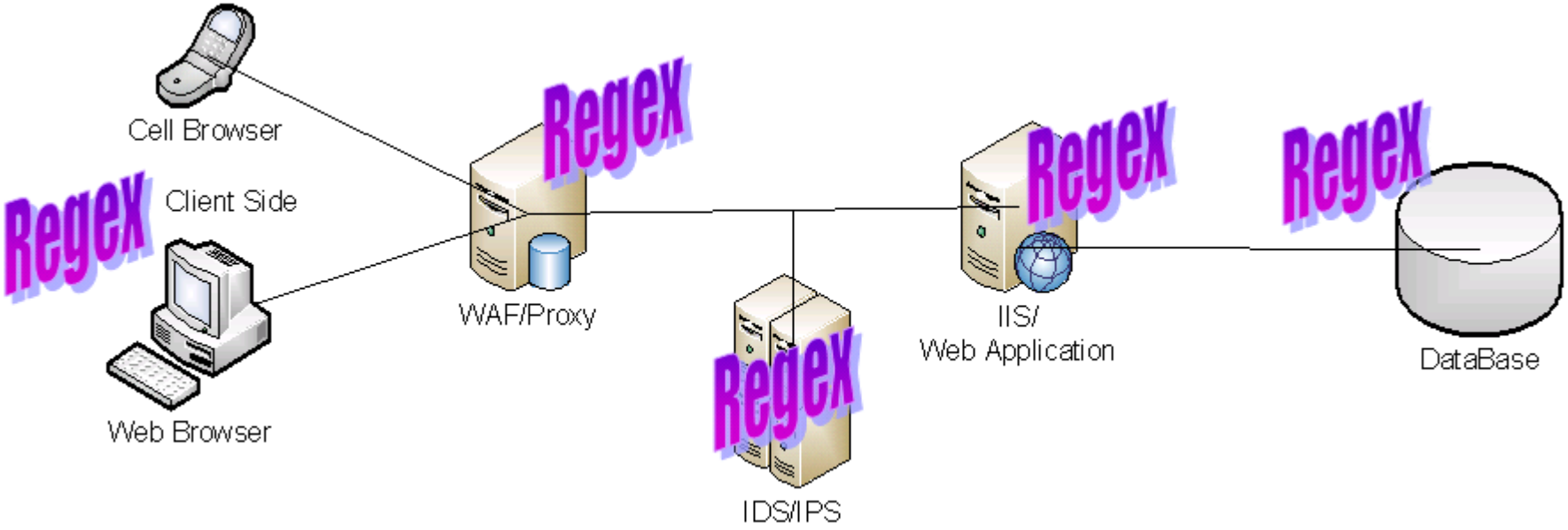
- Requesting a large image size for vector graphics. For instance, SVG and font files.
- Integer overflow errors can cause sanity checking of sizes to fail.
- An object graph constructed by parsing a text or binary stream may have memory requirements many times that of the original data.
- "Zip bombs" whereby a short file is very highly compressed. For instance, ZIPs, GIFs and gzip encoded HTTP contents. When decompressing files, it is better to set limits on the decompressed data size rather than relying upon compressed size or meta-data.
- "Billion laughs attack" whereby XML entity expansion causes an XML document to grow dramatically during parsing. Set the XMLConstants.FEATURE_SECURE_PROCESSING feature to enforce reasonable limits.
- Causing many keys to be inserted into a hash table with the same hash code, turning an algorithm of around O(n) into O(n2).

Secure Coding -DOS

- Regular expressions may exhibit catastrophic backtracking.
- XPath expressions may consume arbitrary amounts of processor time.
- Java deserialization and Java Beans XML deserialization of malicious data may result in unbounded memory or CPU usage.
- Detailed logging of unusual behavior may result in excessive output to log files.
- Infinite loops can be caused by parsing some corner case data. Ensure that each iteration of a loop makes some progress.
- Processing JARs from untrusted sources may lead to resource exhaustion and/or unexpected runtime behavior.

Secure Coding -DOS





Evil Regex

- Vulnerable Regex in online repositories
  - ReGexLib,id=1757 (email validation) - see bold part, which is an Evil Regex
  - ^([a-zA-Z0-9])(([\-.]|[_]+)?([a-zA-Z0-9]+))*(@){1}[a-z0-9]+[.]{1}(([a-z]{2,3})|([a-z]{2,3}[.]{1}[a-z]{2,3}))$
  - Input:
  - aaaaaaaaaaaaaaaaaaaaaaaa!
- OWASP Validation Regex Repository, Java Classname - see bold part, which is an Evil Regex
  - ^(([a-z])+.)+[A-Z]([a-z])+$
  - Input:
  - aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa!

Input Validation

- Guideline 5-1 / INPUT-1: Validate inputs
  - Input from untrusted sources must be validated before use. Maliciously crafted inputs may cause problems, whether coming through method arguments or external streams.
  - Examples include overflow of integer values and directory traversal attacks by including "../" sequences in filenames.
  - Ease-of-use features should be separated from programmatic interfaces.
  - Whenever possible, processing untrusted input should be avoided.
  - For example, consuming a JAR file from an untrusted source might allow an attacker to inject malicious code or data into the system, causing misbehavior, excessive resource consumption, or other problems.

Input Validation

- Guideline 5-2 / INPUT-2: Validate output from untrusted objects as input
  - In general method arguments should be validated but not return values.
  - However, in the case of an upcall (invoking a method of higher level code) the returned value should be validated.
  - Likewise, an object only reachable as an implementation of an upcall need not validate its inputs.
  - A subtle example would be Class objects returned by ClassLoaders.
  - An attacker might be able to control ClassLoader instances that get passed as arguments, or that are set in Thread context.
  - Thus, when calling methods on ClassLoaders not many assumptions can be made.
  - Multiple invocations of ClassLoader.loadClass() are not guaranteed to return the same Class instance or definition, which could cause TOCTOU issues.

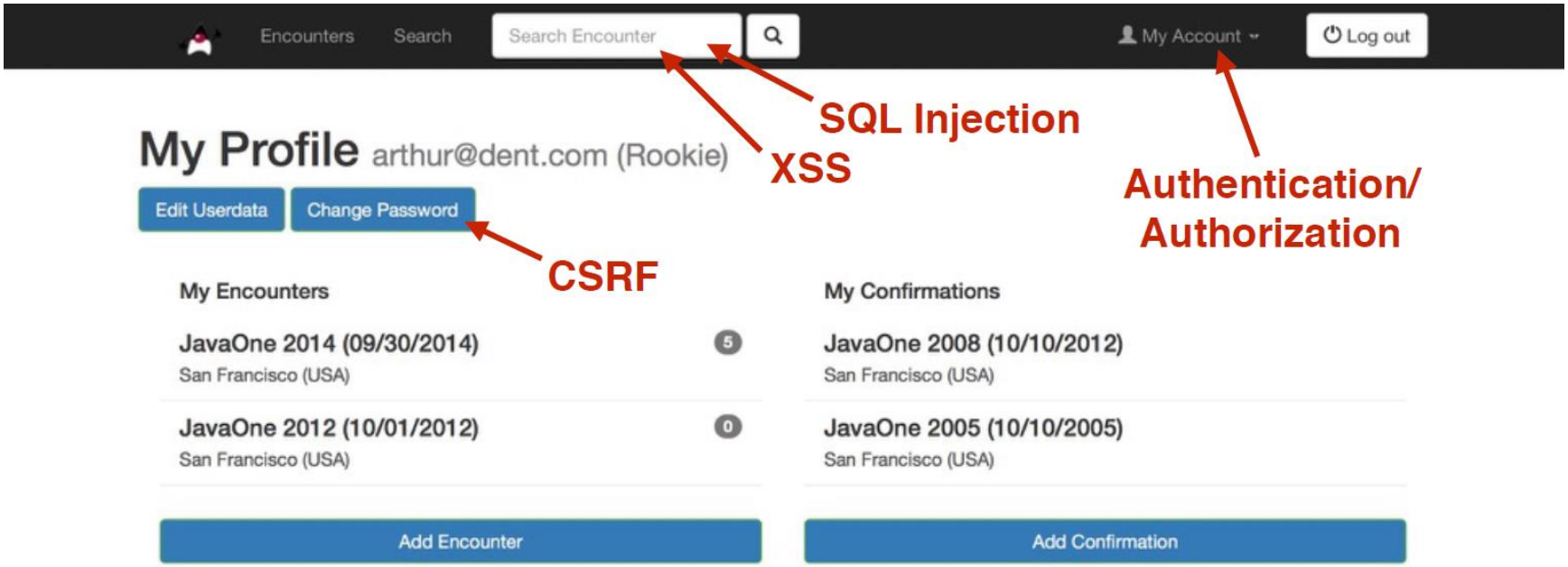
Serialization and Deserialization

- Guideline 8-1 / SERIAL-1: Avoid serialization for security-sensitive classes
  - Making a class serializable effectively creates a public interface to all fields of that class.
  - Serialization also effectively adds a hidden public constructor to a class, which needs to be considered when trying to restrict object construction.
  - Similarly, lambdas should be scrutinized before being made serializable.
  - Functional interfaces should not be made serializable without due consideration for what could be exposed.
  - It is also important to avoid unintentionally making a security-sensitive class serializable, either by subclassing a serializable class or implementing a serializable interface.

Serialization and Deserialization

- Guideline 8-2 / SERIAL-2: Guard sensitive data during serialization
  - Once an object has been serialized the Java language's access controls can no longer be enforced and attackers can access private fields in an object by analyzing its serialized byte stream. Therefore, do not serialize sensitive data in a serializable class.
  - Approaches for handling sensitive fields in serializable classes are:
    - Declare sensitive fields transient
    - Define the serialPersistentFields array field appropriately
    - Implement writeObject and use ObjectOutputStream.putField selectively
    - Implement writeReplace to replace the instance with a serial proxy
    - Implement the Externalizable interface

Threat Model



Threat Model

- Security flaws are introduced early in the development lifecycle, with no code developed yet.
- Threat modeling is all about finding security problems.
- Threat modeling starts early

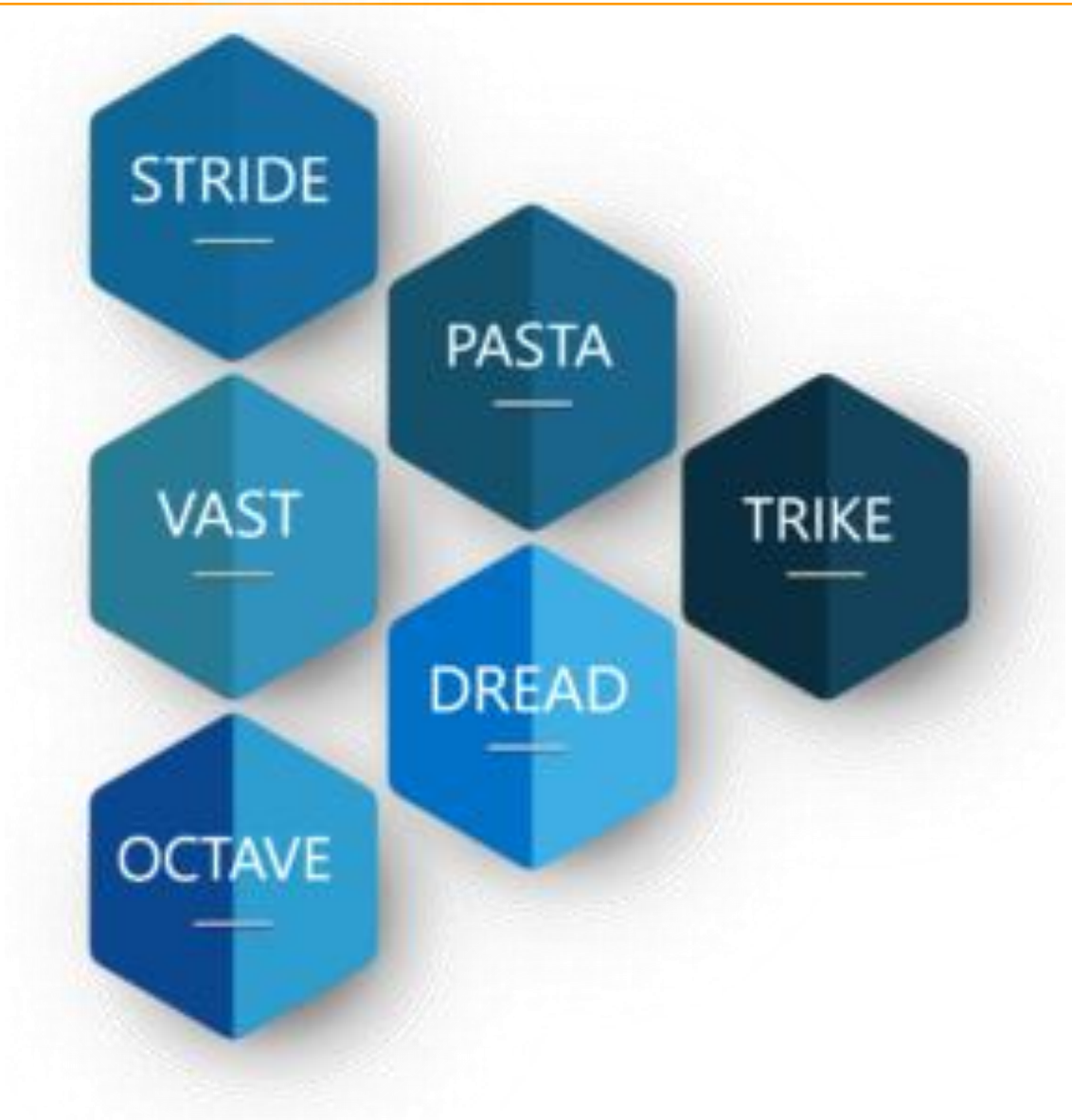
Different ways to threat model

- Focus on attackers: Can you really think like an attacker?
- Focus on assets: What is an asset in your application? How do you link assets to threats?

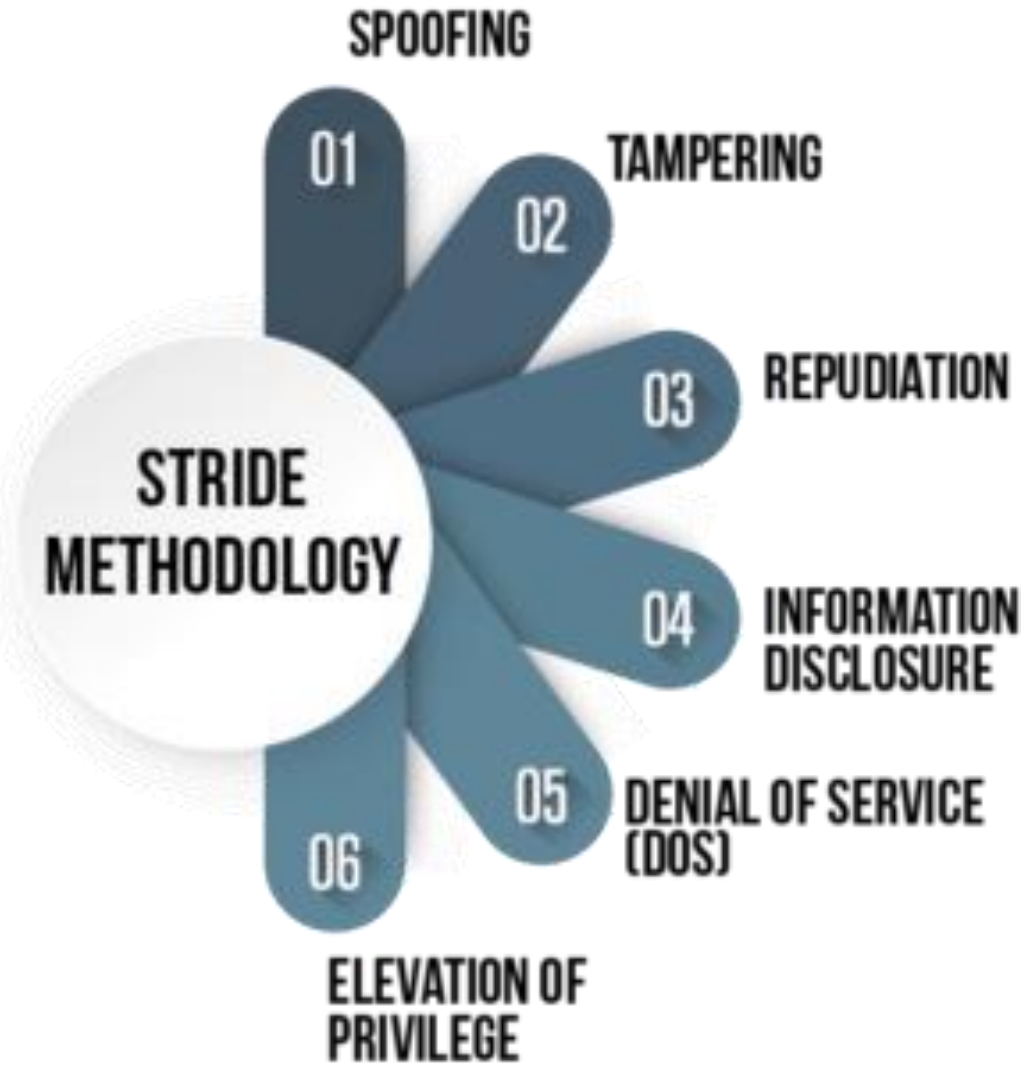
Threat Model

STEPS TO
THREAT MODELING

Threat Model Methodologies



Threat Model STRIDE



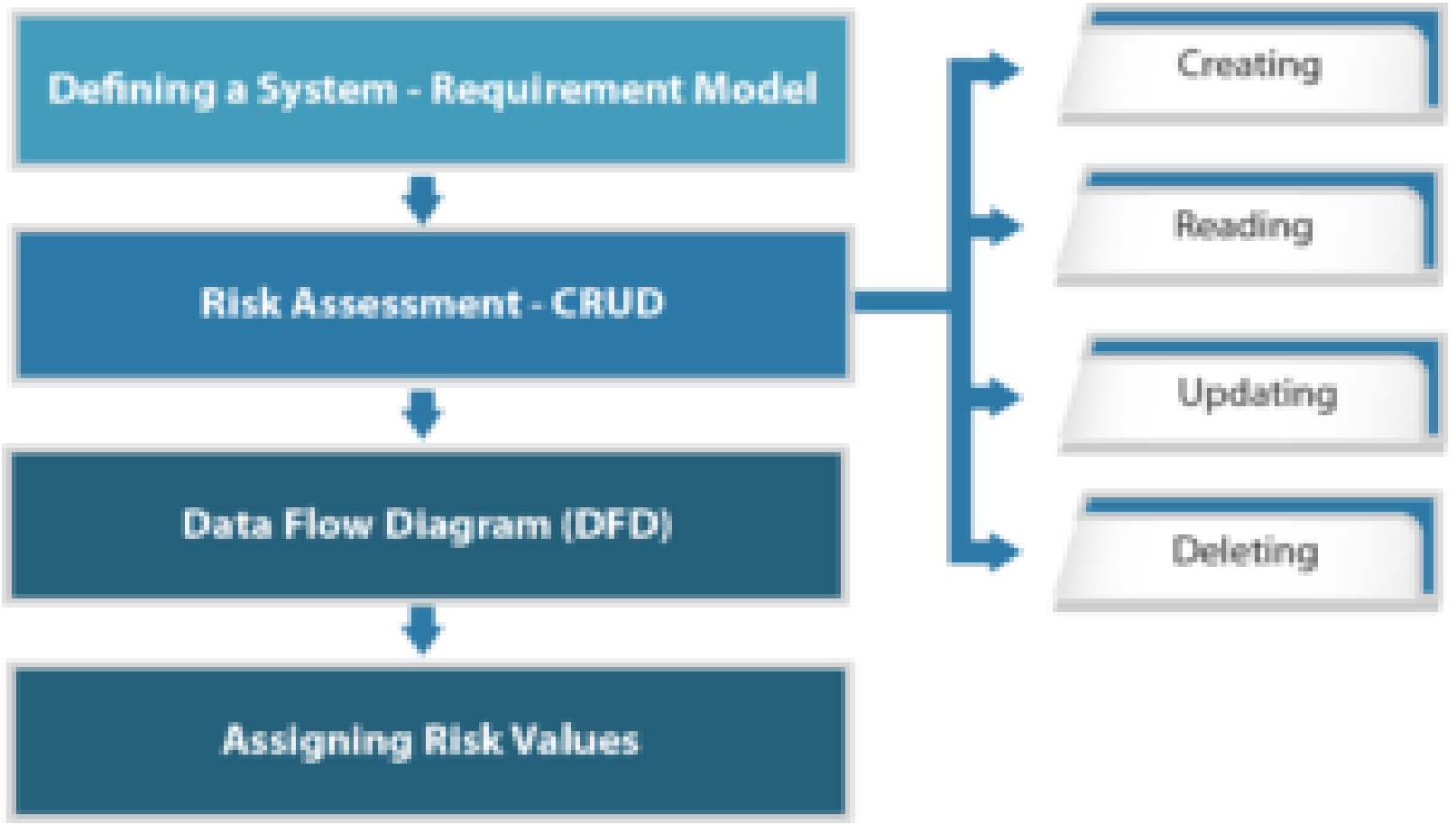
Threat Model STRIDE

	Type of Threat	What Was Violated	How Was It Violated?
S	Spoofing	Authentication	Impersonating something or someone known and trusted.
T	Tampering	Integrity	Modifying data on disk, memory, network, etc.,
R	Repudiation	Non-repudiation	Claim to not be responsible for an action
I	Information Disclosure	Confidentiality	Providing information to someone who is not authorized
D	Denial of Service (DoS)	Availability	Denying or obstructing access to resources required to provide service
E	Elevation of Privilege	Authorization	Allowing access to someone without proper authorization

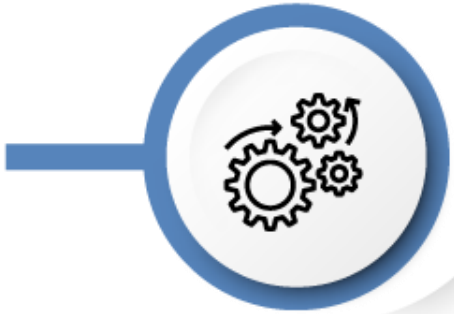
Stages of Process for Attack Simulation and Threat Analysis (PASTA)

Stages of Process for Attack Simulation and Threat Analysis (PASTA)

TRIKE

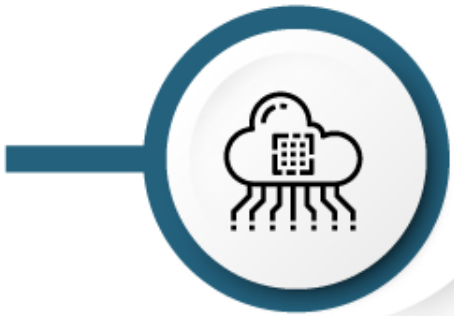

Defining a System - Requirement Model
Risk Assessment - CRUD
Data Flow Diagram (DFD)
Assigning Risk Values
Creating
Reading
Updating
Deleting

VAST



Automation

- Eliminates Repetition in Threat Modeling
- Ongoing Threat Modeling
- Scaled to Encompass the Entire Enterprise



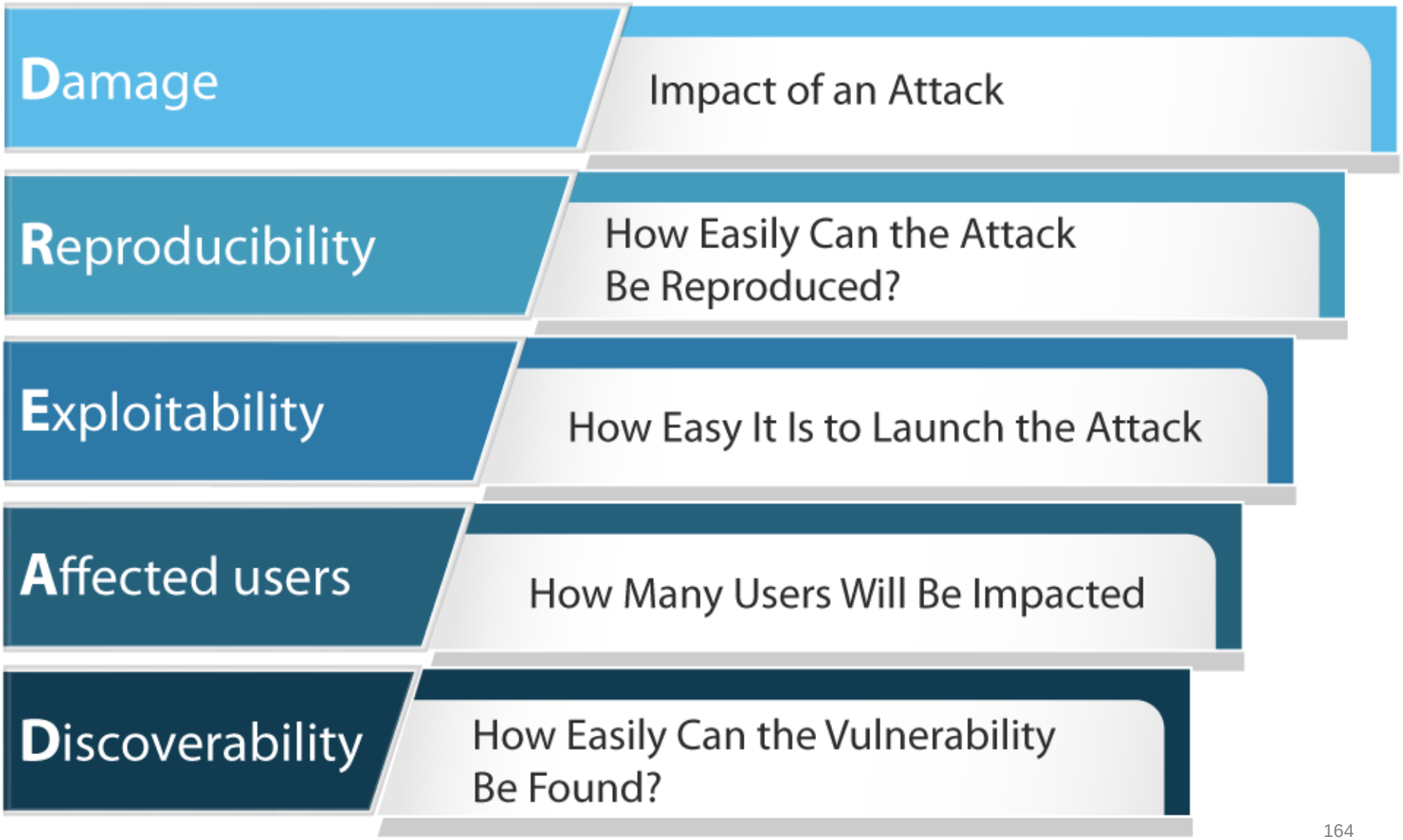
Integration

- Integration with Tools Throughout the SDLC
- Supports the Agile DevOps

Collaboration

Key Stakeholders Collaboration: App Developers, Systems Architects, Security Team, and Senior Executives

DREAD


Damage
Impact of an Attack
Reproducibility
How Easily Can the Attack
Be Reproduced?
Exploitability
How Easy It Is to Launch the Attack
Affected users
How Many Users Will Be Impacted
Discoverability
How Easily Can the Vulnerability
Be Found?

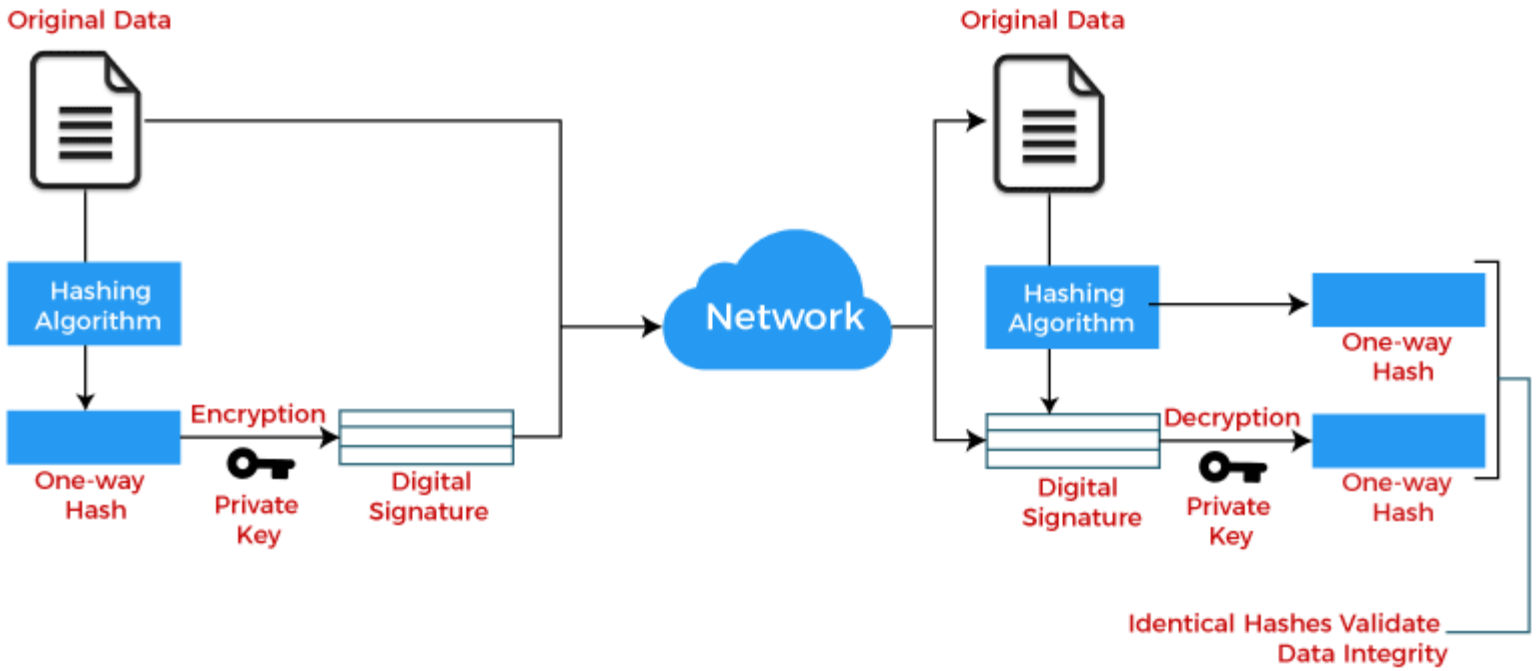
OCTAVE


Identify Risk Evaluation Criteria
Identify Assets That Are Crucial for the Objective of the Organization
Evaluate the Potential Consequences of These Threats to the Organization
Understand the Organization's Operational Risk Tolerance
Identify Threats and Vulnerabilities to Those Assets
Initiate Actions to Mitigate Risks

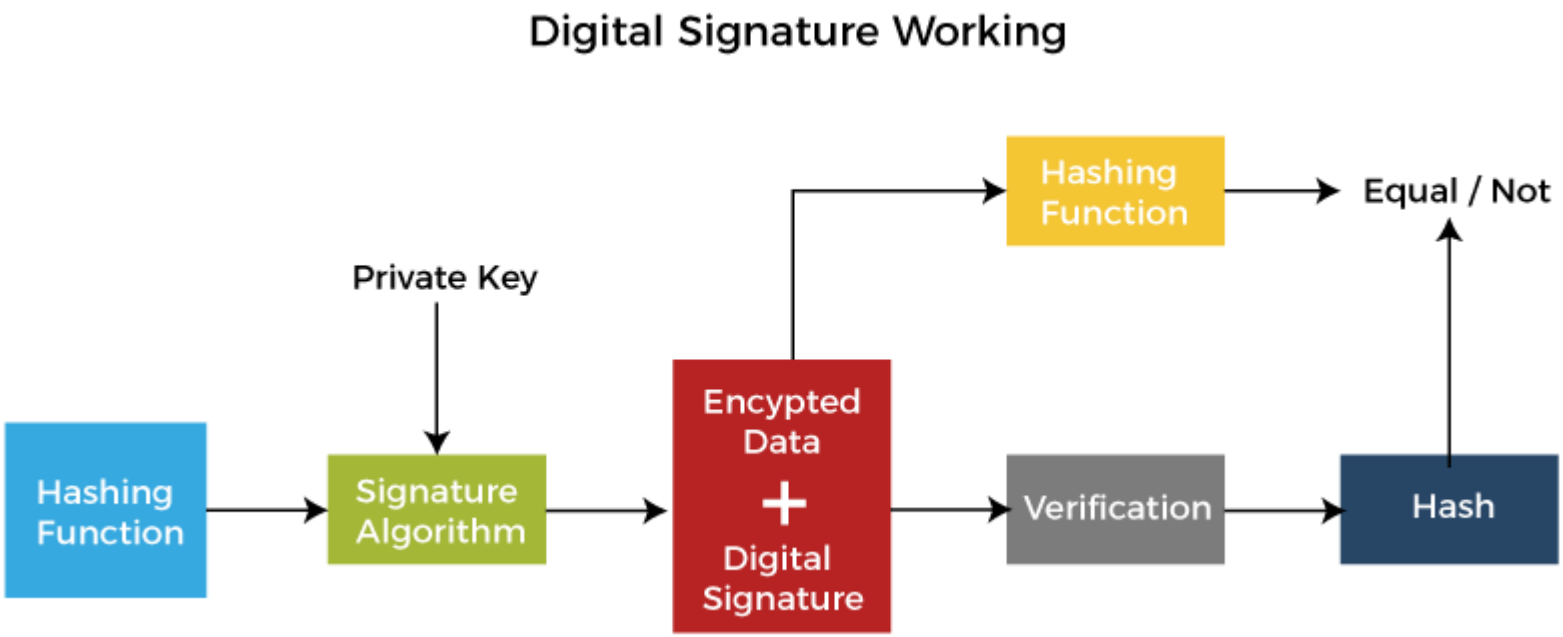
Threat Modelling Tools

- Microsoft Threat Modelling Tool
- ThreatModeler
- securiCAD Professional
- IriusRisk
- Tutamen
- OWASP Threat Dragon

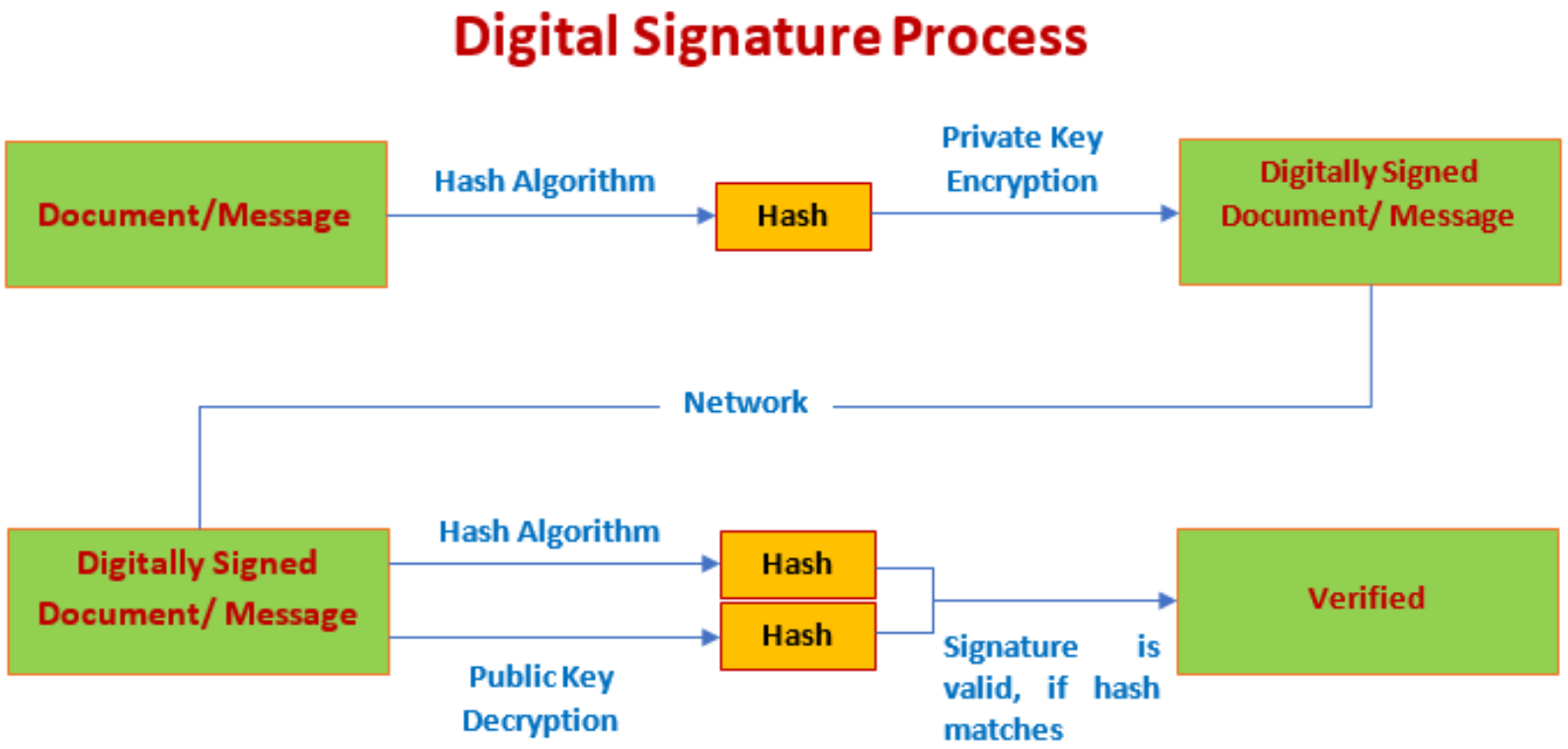
Digital Signature



Digital Signature



Digital Signature



Java SE Security Mechanisms

High-level Features	Low-level Features	Benefits
Platform Security	Built-in language security features enforced by the Java compiler and virtual machine: •Strong data typing •Automatic memory management •Bytecode verification •Secure class loading	Provides a safe and secure platform for developing and running applications. Compile-time data type checking and automatic memory management leads to more robust code and reduces memory corruption and vulnerabilities. Bytecode verification ensures code conforms to the JVM specification and prevents hostile code from corrupting the runtime environment. Class loaders ensure that untrusted code cannot interfere with the running of other Java programs.

Java SE Security Mechanisms

High-level Features	Low-level Features	Benefits
Cryptography	•Comprehensive API with support for a wide range of cryptographic services including digital signatures, message digests, ciphers (symmetric, asymmetric, stream & block), message authentication codes, key generators and key factories •Support for a wide range of standard algorithms including RSA, DSA, AES, Triple-DES, SHA, PKCS5, RC2, and RC4. •PKCS#11 cryptographic token support	Provides an extensible, full featured API for building secure applications: •Algorithm and implementation independent •Provider-based (pluggable) architecture

Java SE Security Mechanisms

High-level Features	Low-level Features	Benefits
Authentication and Access Control	•Abstract authentication APIs that can incorporate a wide range of login mechanisms through a pluggable architecture. •A comprehensive policy and permissions API that allows the developer to create and administer applications requiring fine-grained access to security-sensitive resources.	Enables single sign-on of multiple authentication mechanisms and fine-grained access to resources based on the identity of the user or code signer. Support for timestamped signatures makes it easier to deploy signed code by avoiding the need to re-sign code when the signer's certificate expires.

Java SE Security Mechanisms

High-level Features	Low-level Features	Benefits
Secure Communications	•JSSE standard API, available in the javax.net and javax.net.ssl packages, which provides secure sockets for client and server-side applications, a non-blocking engine for producing and consuming streams of TLS/DTLS data, and key and trust manager interfaces.	Authenticates peers over an untrusted network and protects the integrity and privacy of data transmitted between them.

Java SE Security Mechanisms

High-level Features	Low-level Features	Benefits
Secure Communications	•A JSSE provider named SunJSSE that provides cryptographic services including: • SSL 3.0, TLS (versions 1.0, 1.1, 1.2, and 1.3), and DTLS (versions 1.0 and 1.2) security protocols • Implementations of TLS and DTLS cipher suites, an X.509-based key manager and an X.509-based trust manager. •.	Authenticates peers over an untrusted network and protects the integrity and privacy of data transmitted between them.

Java SE Security Mechanisms

High-level Features	Low-level Features	Benefits
Secure Communications	•The Java GSS-API, which provides uniform access to security services on a variety of underlying security mechanisms, including Kerberos. •The Java Simple Authentication and Security Layer (SASL), which specifies a protocol for authentication and optional establishment of a security layer between client and server applications	Authenticates peers over an untrusted network and protects the integrity and privacy of data transmitted between them.

Java SE Security Mechanisms

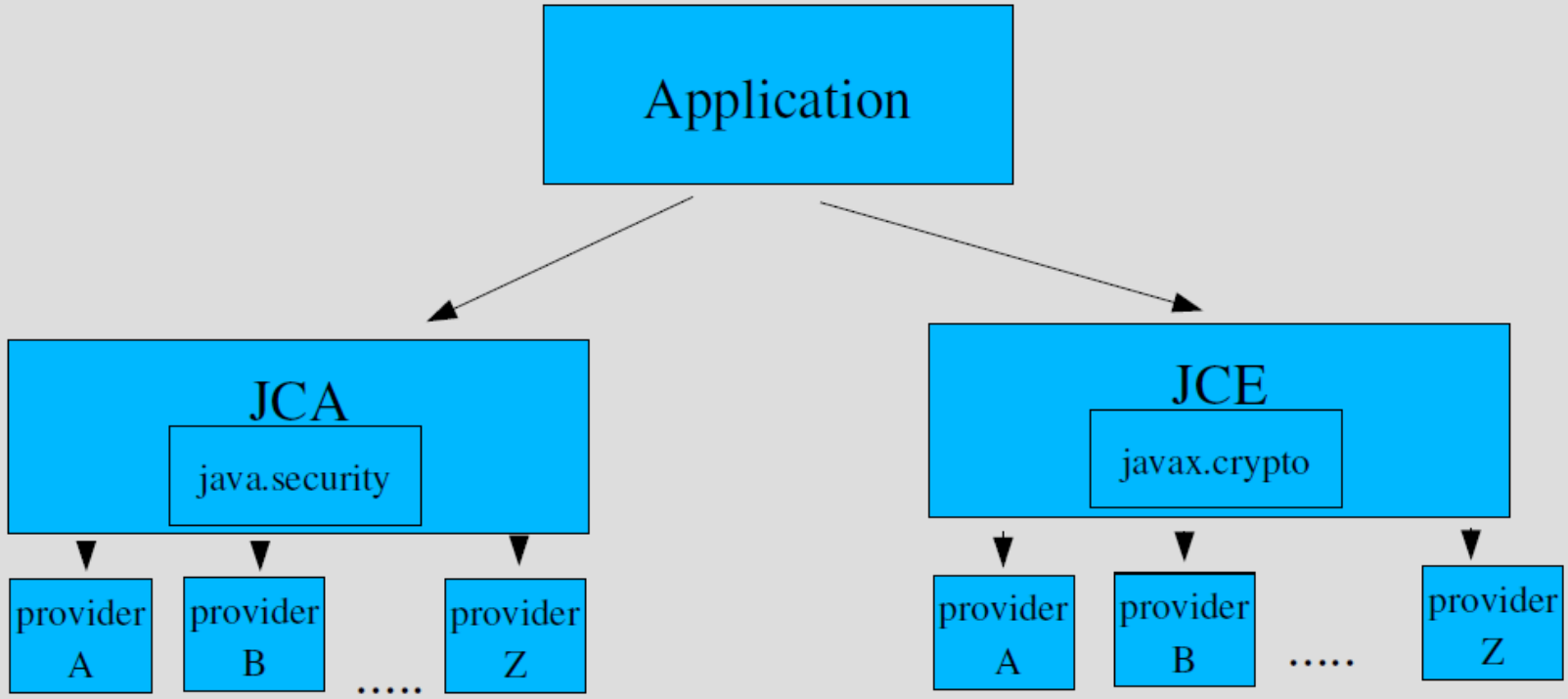
High-level Features	Low-level Features	Benefits
Public Key Infrastructure (PKI)	Tools for managing keys and certificates and comprehensive, abstract APIs with support for the following features and algorithms: •Certificates and Certificate Revocation Lists (CRLs): X.509 •Certification Path Validators and Builders: PKIX, Online Certificate Status Protocol (OCSP) •KeyStores: PKCS#11, PKCS#12 •Certificate Stores (Repositories): LDAP, java.util.Collection	Eases the development and deployment of complex PKI applications. Support for OCSP provides a more scalable and timely method for applications to check certificate revocation status.

Java Cryptography Architecture

- The Java Cryptography Architecture (JCA) is a major piece of the platform, and contains
- a "provider" architecture and a set of APIs for digital signatures
- message digests (hashes)
- certificates and certificate validation
- encryption (symmetric/asymmetric block/stream ciphers)
- key generation and management, and secure random number generation.

Java Cryptography Architecture

- Provider based architecture



Java Cryptography Design Principles

Algorithm independence

– specification of engine classes

● Algorithm extensibility

– easy updation of engine classes with new algorithms

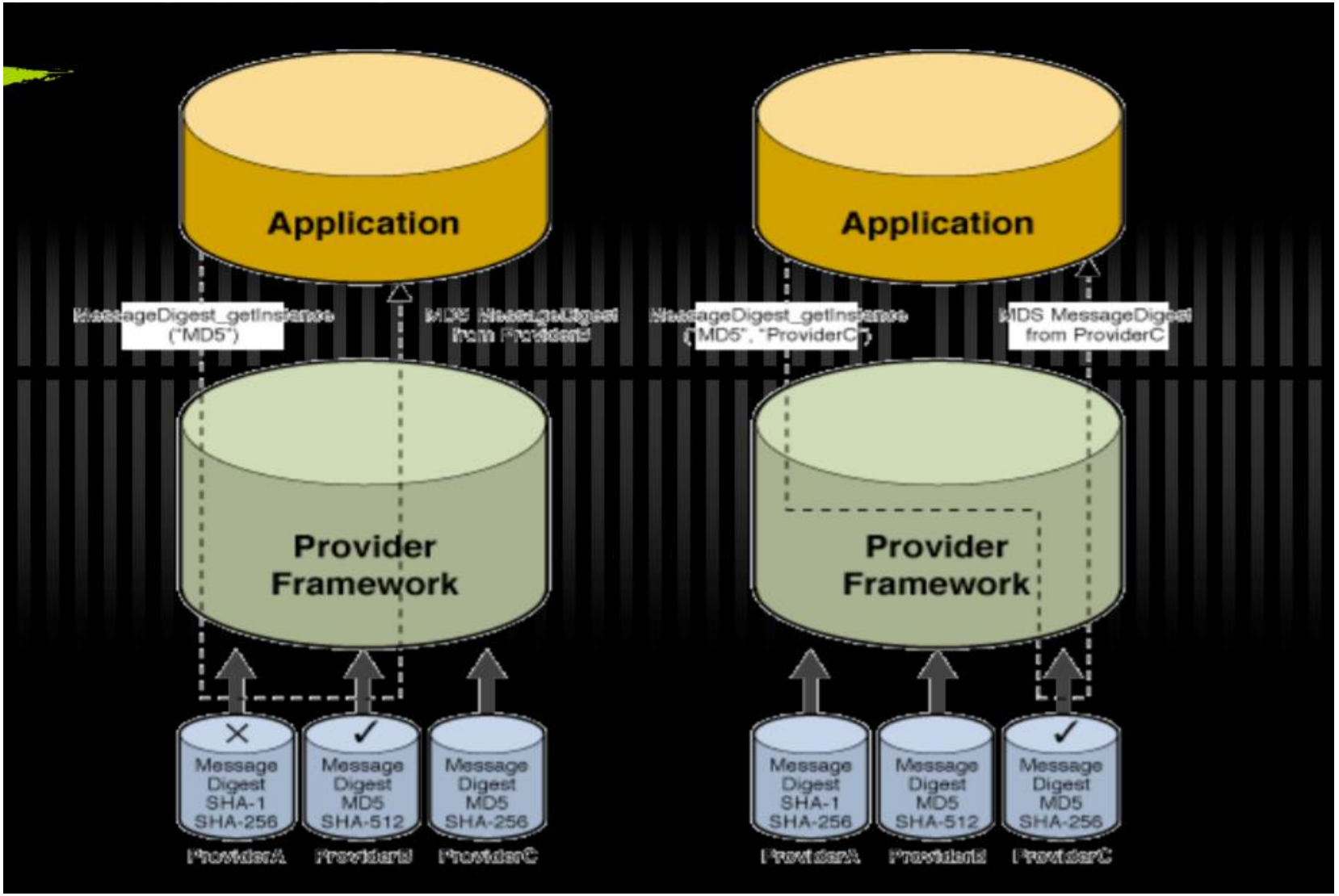
● Implementation independence

– use of cryptographic service providers

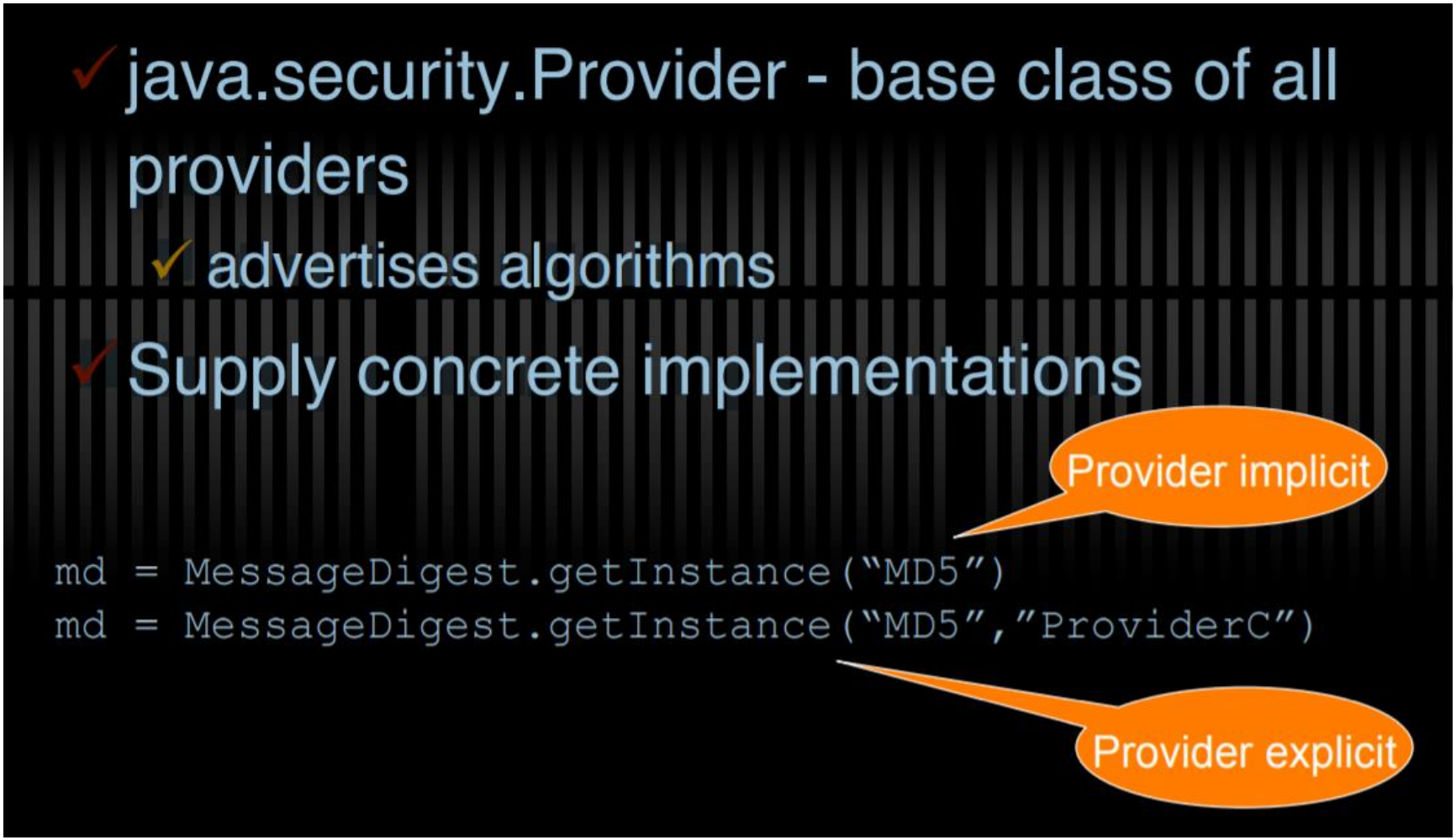
● Implementation interoperability

– providers working with each other

Java Cryptography Architecture



Providers


java.security.Provider - base class of all providers
advertises algorithms
Supply concrete implementations
Provider implicit
md = MessageDigest.getInstance("MD5")
md = MessageDigest.getInstance("MD5","ProviderC")
Provider explicit

JCA Engine Classes

- Provide the interface to a specific type of cryptographic service
  - cryptographic operatoins
  - generators or converers of cryptographic material
  - objects (keystores or certificates)

JCA Engine Classes

MessageDigest (produces hash value)

- Signature (produces digital signature)

– KeyPairGenerator (produces pair of keys)

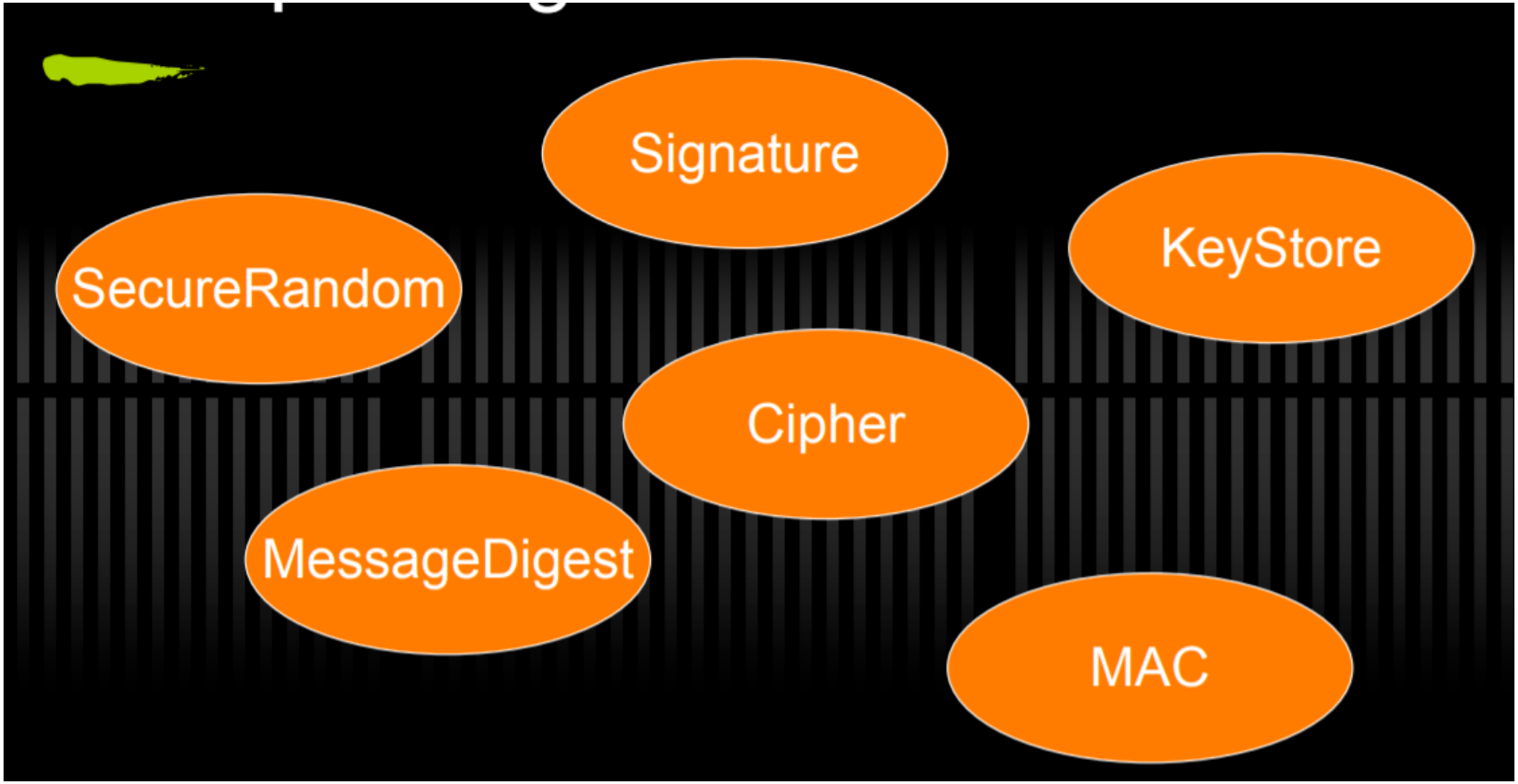
- KeyFactory (breaks down a key)
- KeyStore (manages and stores keys)
- SecureRandom (produces random numbers)

– AlgorithmParameters (encoding and decoding)

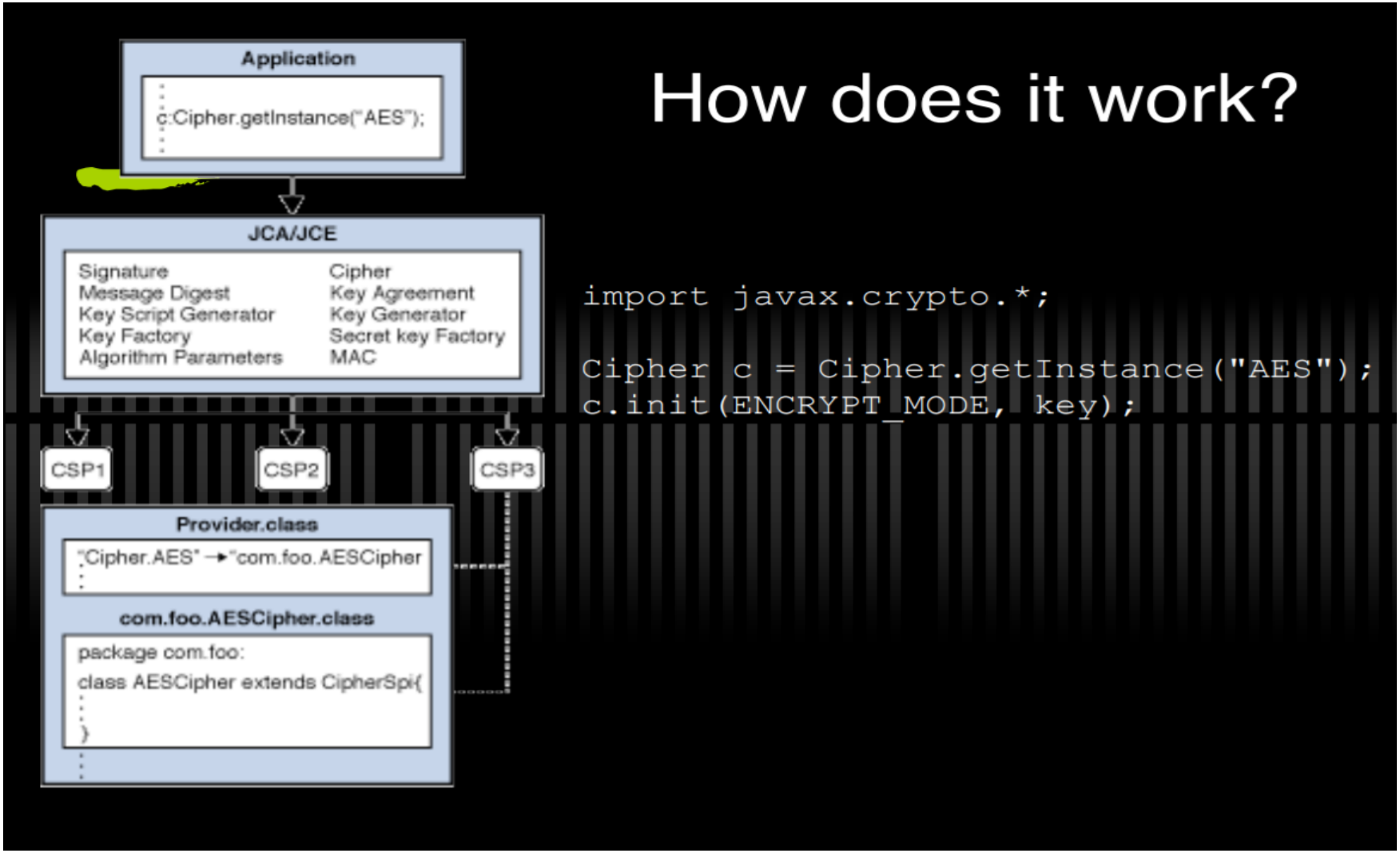
– AlgorithmParameterGenerator (generates parameters)

- CertificateFactory (public key cert, revocation)
- CertPathBuilder (establish relationship chains between certs)
- CertStore (stores certificates and revocation lists)

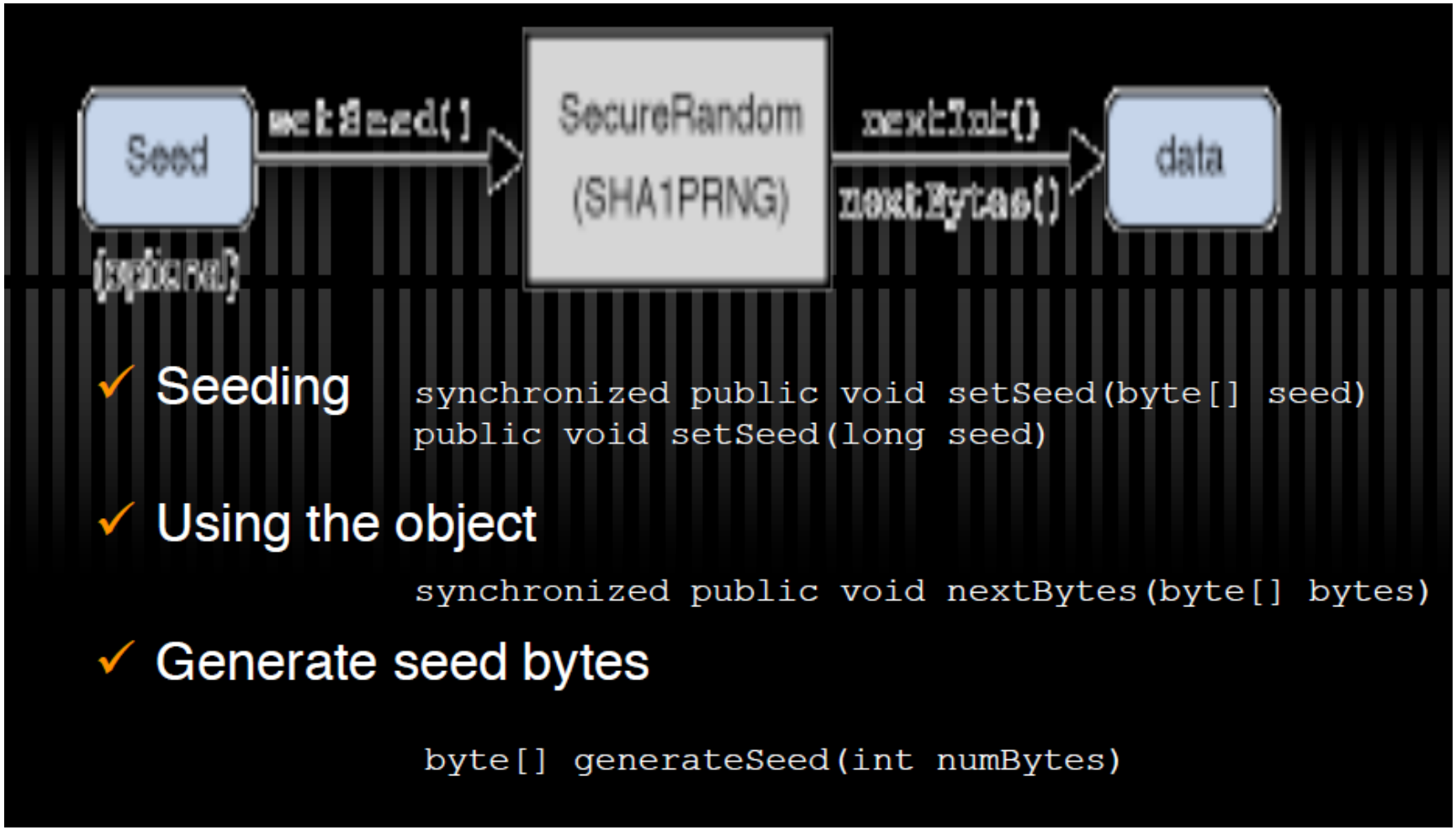
JCA Engine Classes



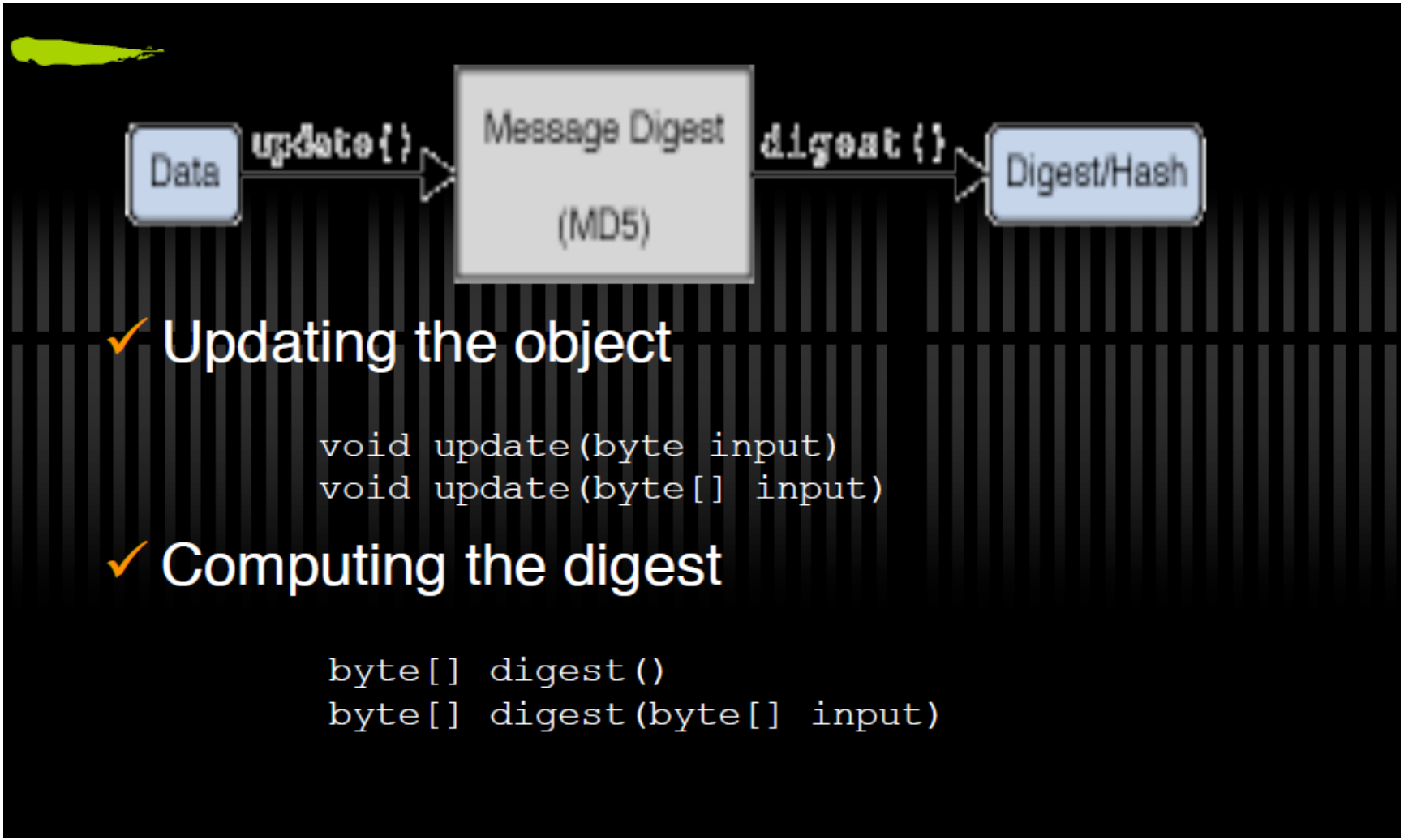
JCA Engine Classes



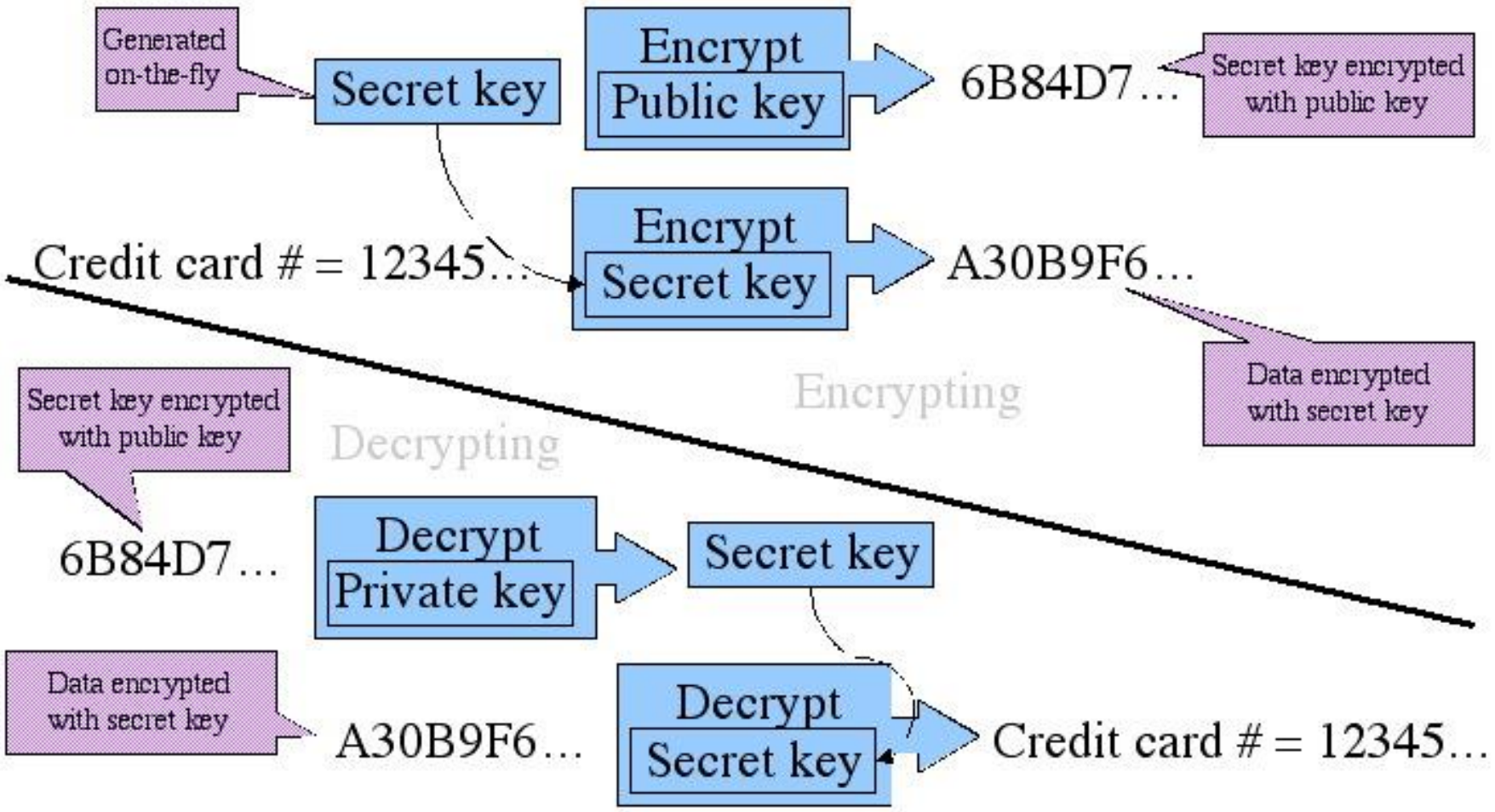
Secure Random



Message Digest

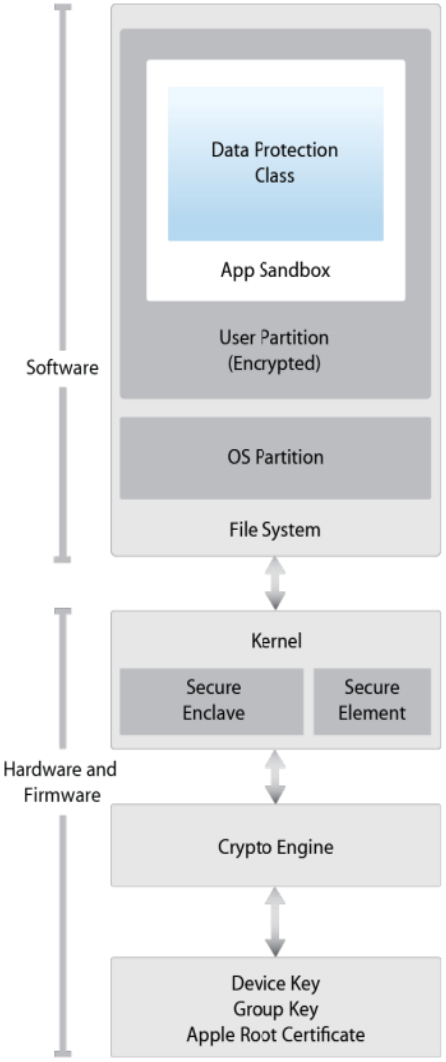


Java Cryptography Extension



iOS Security Architecture

- The iOS security architecture, officially documented by Apple in the iOS Security Guide, consists of six core features. This security guide is updated by Apple for each major iOS version:
- Hardware Security
- Secure Boot
- Code Signing
- Sandbox
- Encryption and Data Protection
- General Exploit Mitigations



iOS Security Architecture

- Hardware Security
  - Each iOS device comes with two built-in Advanced Encryption Standard (AES) 256-bit keys.
  - The device’s unique IDs (UIDs) and a device group IDs (GIDs) are AES 256-bit keys fused (UID) or compiled (GID) into the Application Processor (AP) and Secure Enclave Processor (SEP) during manufacturing.
  - There's no direct way to read these keys with software or debugging interfaces such as JTAG.
  - Encryption and decryption operations are performed by hardware AES crypto-engines that have exclusive access to these keys.

iOS Security Architecture

- Hardware Security
  - The GID is a value shared by all processors in a class of devices used to prevent tampering with firmware files and other cryptographic tasks not directly related to the user's private data.
  - UIDs, which are unique to each device, are used to protect the key hierarchy that's used for device-level file system encryption. Because UIDs aren't recorded during manufacturing, not even Apple can restore the file encryption keys for a particular device.

iOS Security Architecture

- Hardware Security
  - To allow secure deletion of sensitive data on flash memory, iOS devices include a feature called Effaceable Storage.
  - This feature provides direct low-level access to the storage technology, making it possible to securely erase selected blocks.

iOS Security Architecture

- Secure Boot
  - When an iOS device is powered on, it reads the initial instructions from the read-only memory known as Boot ROM, which bootstraps the system.
  - The Boot ROM contains immutable code and the Apple Root CA, which is etched into the silicon chip during the fabrication process, thereby creating the root of trust.
  - Next, the Boot ROM makes sure that the LLB's (Low Level Bootloader) signature is correct, and the LLB checks that the iBoot bootloader's signature is correct too.
  - After the signature is validated, the iBoot checks the signature of the next boot stage, which is the iOS kernel.
  - If any of these steps fail, the boot process will terminate immediately and the device will enter recovery mode and display the restore screen.
  - However, if the Boot ROM fails to load, the device will enter a special low-level recovery mode called Device Firmware Upgrade (DFU).

iOS Security Architecture

- Secure Boot
  - This is the last resort for restoring the device to its original state.
  - In this mode, the device will show no sign of activity; i.e., its screen won't display anything.
  - This entire process is called the "Secure Boot Chain". Its purpose is focused on verifying the boot process integrity, ensuring that the system and its components are written and distributed by Apple.
  - The Secure Boot chain consists of the kernel, the bootloader, the kernel extension, and the baseband firmware.

iOS Security Architecture

- Code Signing
  - Apple has implemented an elaborate DRM system to make sure that only Apple-approved code runs on their devices, that is, code signed by Apple.
  - In other words, you won't be able to run any code on an iOS device that hasn't been jailbroken unless Apple explicitly allows it.
  - End users are supposed to install apps through the official Apple's App Store only.
  - For this reason (and others), iOS has been compared to a crystal prison.
  - A developer profile and an Apple-signed certificate are required to deploy and run an application

iOS Security Architecture

- Code Signing
  - . Developers need to register with Apple, join the Apple Developer Program and pay a yearly subscription to get the full range of development and deployment possibilities.
  - There's also a free developer account that allows you to compile and deploy apps (but not distribute them in the App Store) via sideloading.

iOS Security Architecture

- Encryption and Data Protection
  - FairPlay Code Encryption is applied to apps downloaded from the App Store.
  - FairPlay was developed as a DRM when purchasing multimedia content.
  - Originally, FairPlay encryption was applied to MPEG and QuickTime streams, but the same basic concepts can also be applied to executable files.
  - The basic idea is as follows: Once you register a new Apple user account, or Apple ID, a public/private key pair will be created and assigned to your account.

iOS Security Architecture

- Encryption and Data Protection
  - The private key is securely stored on your device. This means that FairPlay-encrypted code can be decrypted only on devices associated with your account.
  - Reverse FairPlay encryption is usually obtained by running the app on the device, then dumping the decrypted code from memory (see also "Basic Security Testing on iOS").

iOS Security Architecture

- Encryption and Data Protection
  - Apple has built encryption into the hardware and firmware of its iOS devices since the release of the iPhone 3GS.
  - Every device has a dedicated hardware-based cryptographic engine that provides an implementation of the AES 256-bit encryption and the SHA-1 hashing algorithms.
  - In addition, there's a unique identifier (UID) built into each device's hardware with an AES 256-bit key fused into the Application Processor.
  - This UID is unique and not recorded elsewhere. At the time of writing, neither software nor firmware can directly read the UID.
  - Because the key is burned into the silicon chip, it can't be tampered with or bypassed. Only the crypto engine can access it.

iOS Security Architecture

- Encryption and Data Protection
  - Building encryption into the physical architecture makes it a default security feature that can encrypt all data stored on an iOS device.
  - As a result, data protection is implemented at the software level and works with the hardware and firmware encryption to provide more security.

iOS Security Architecture

- Sandbox
  - The appsandbox is an iOS access control technology. It is enforced at the kernel level.
  - Its purpose is limiting system and user data damage that may occur when an app is compromised.
  - Sandboxing has been a core security feature since the first release of iOS.
  - All third-party apps run under the same user (mobile), and only a few system applications and services run as root (or other specific system users).
  - Regular iOS apps are confined to a container that restricts access to the app's own files and a very limited number of system APIs.
  - Access to all resources (such as files, network sockets, IPCs, and shared memory) are controlled by the sandbox.

iOS Security Architecture

- Sandbox
  - These restrictions work as follows [#levin]:
  - The app process is restricted to its own directory (under /var/mobile/Containers/Bundle/Application/ or /var/containers/Bundle/Application/, depending on the iOS version) via a chroot-like process.
  - The mmap and mmprotect system calls are modified to prevent apps from making writable memory pages executable and stopping processes from executing dynamically generated code.
  - In combination with code signing and FairPlay, this strictly limits what code can run under specific circumstances (e.g., all code in apps distributed via the App Store is approved by Apple).

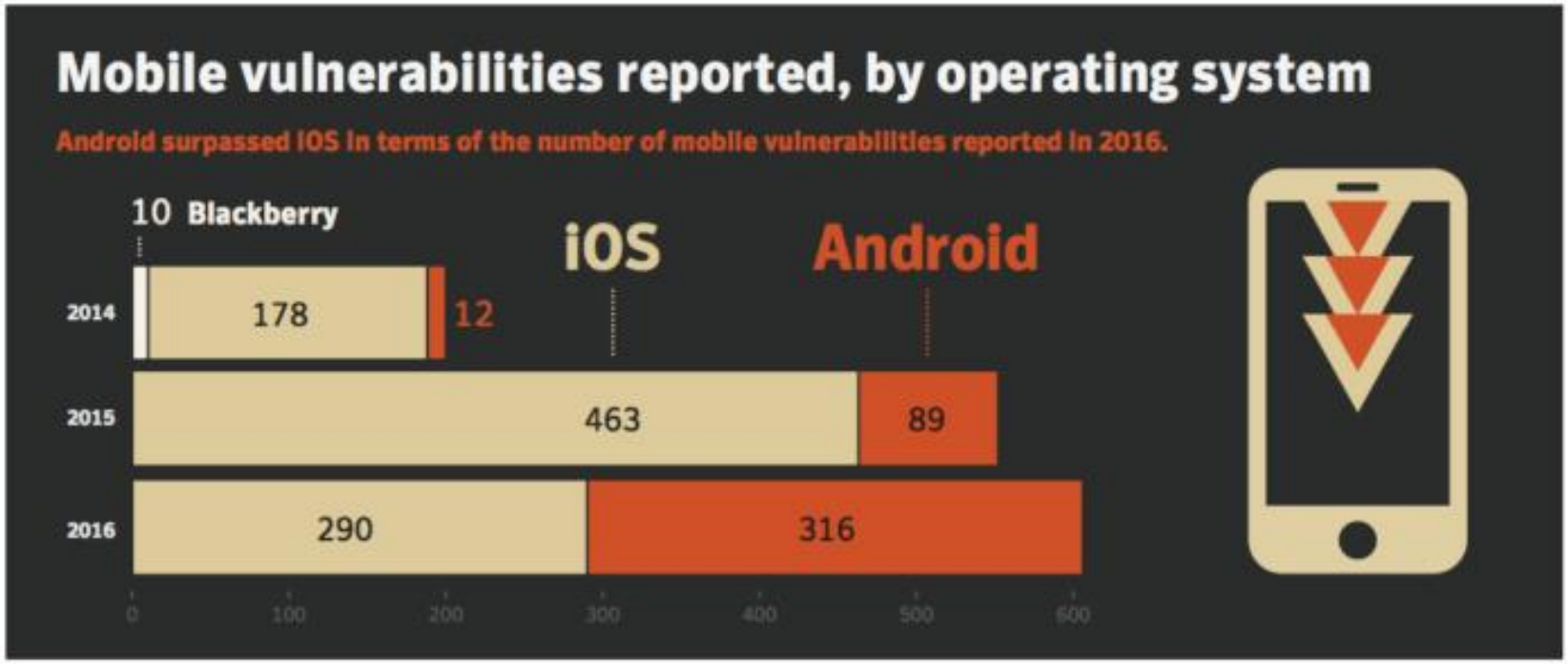
iOS Security Architecture

- Sandbox
  - Processes are isolated from each other, even if they are owned by the same UID at the operating system level.Hardware drivers can't be accessed directly. Instead, they must be accessed through Apple's public frameworks.

iOS Security Architecture

- General Exploit Mitigations
  - iOS implements address space layout randomization (ASLR) and eXecute Never (XN) bit to mitigate code execution attacks.
  - ASLR randomizes the memory location of the program's executable file, data, heap, and stack every time the program is executed.
  - Because the shared libraries must be static to be accessed by multiple processes, the addresses of shared libraries are randomized every time the OS boots instead of every time the program is invoked.
  - This makes specific function and library memory addresses hard to predict, thereby preventing attacks such as the return-to-libc attack, which involves the memory addresses of basic libc functions.

Android vs iOS security: Which is better?



Android vs iOS security: Which is better?

Top mobile threats in 2016

The two most commonly seen mobile malware detections are generic detection names, used to block a wide range of unclassified Android threats.

Mobile Threat	Percentage
Android.Malapp	39.2
Android.MalDownloader	16.1
Android.Opfake	5.2
Android.HiddenAds	4.8
Android.Premiumtext	4.1
Android.Maldropper	2.1
Android.Mobilespy	1.9
Android.Downloader	1.7
Android.Dropper	1.7
Android.Fakeapp	1.7
Android.Smsstealer	1.7
Android.Rootnik	1.6
Android.Lotoor	1.4
Android.SmsBlocker	1.4
Android.MobileSpy	1.3
Android.RegSMS	1.2
Android.FakeInst	1.2
Android.SMSblocker	0.9
Android.HiddenApp	0.8
Android.Lockdroid.E	0.8

Spring Security

- Spring framework has two major applications which are authentication and authorization.
- Authentication is a process of identifying and knowing the target user which wants to access the application.
- Authorization is a process of allowing the authority to perform some actions in the application.
- Apply authorization to a web request, methods and access to an individual domain.

Spring Security

- Spring framework has two major applications which are authentication and authorization.
- Authentication is a process of identifying and knowing the target user which wants to access the application.
- Authorization is a process of allowing the authority to perform some actions in the application.
- Apply authorization to a web request, methods and access to an individual domain.

General Security Principles

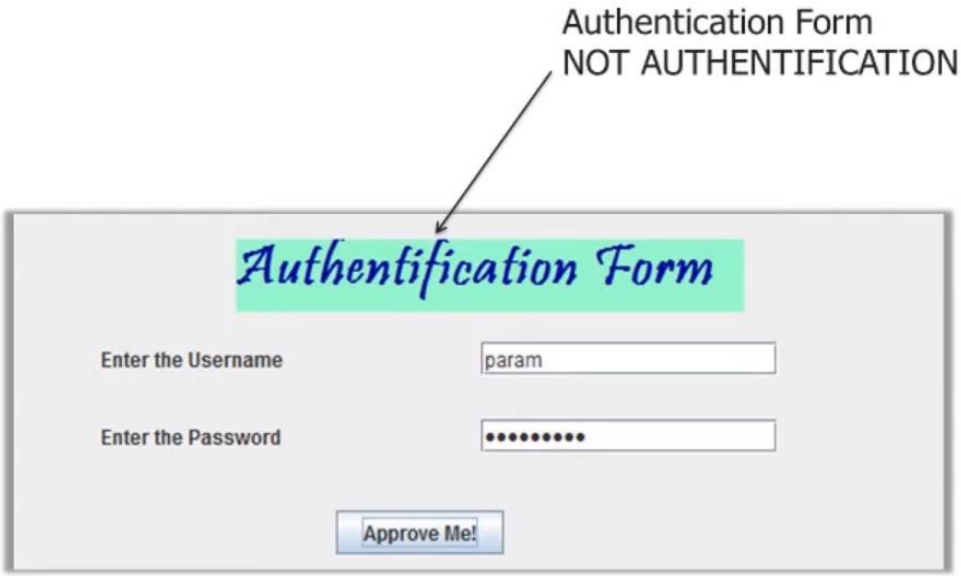
→ Identification: Identifies a user based upon a certain identity; such as a username, token or certificate

→ Authentication: The process of verifying the presented identity. In case of a username it is as simple as checking the supplied password

→ Authorization: The process of identifying the functionality an authenticated user is allowed to use

Authentication

Authentication



Authenticating User

What is Authentication in Spring Security

→ Let's consider a standard authentication scenario that everyone is familiar with

- » A user is prompted to log in with a username and password
- » The system successfully verifies that the password is correct for the username
- » The context information for that user is obtained (their list of roles and so on)
- » A security context is established for the user
- » The user proceeds, potentially to perform some operation which is potentially protected by an access control mechanism which checks the required permissions for the operation against the current security context information

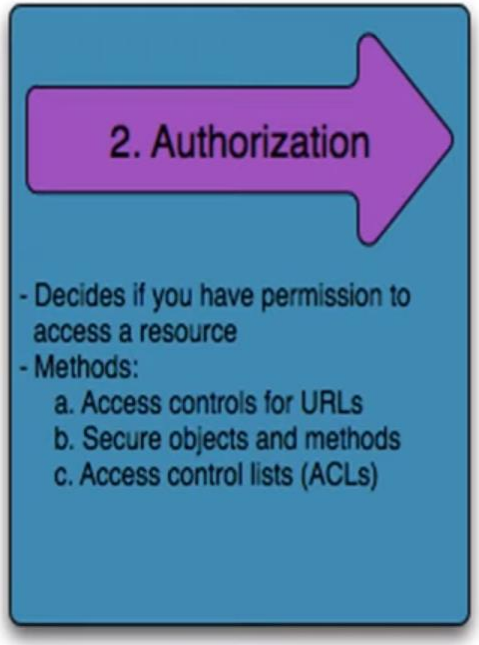
What is Authorization in Spring Security

→ Spring Security tackles security from two angles

→ To secure web requests and restrict access at the URL level, Spring Security uses servlet filters

→ Spring Security can also secure method invocations using Spring AOP—proxying objects and applying advice that ensures that the user has proper authority to invoke secured methods

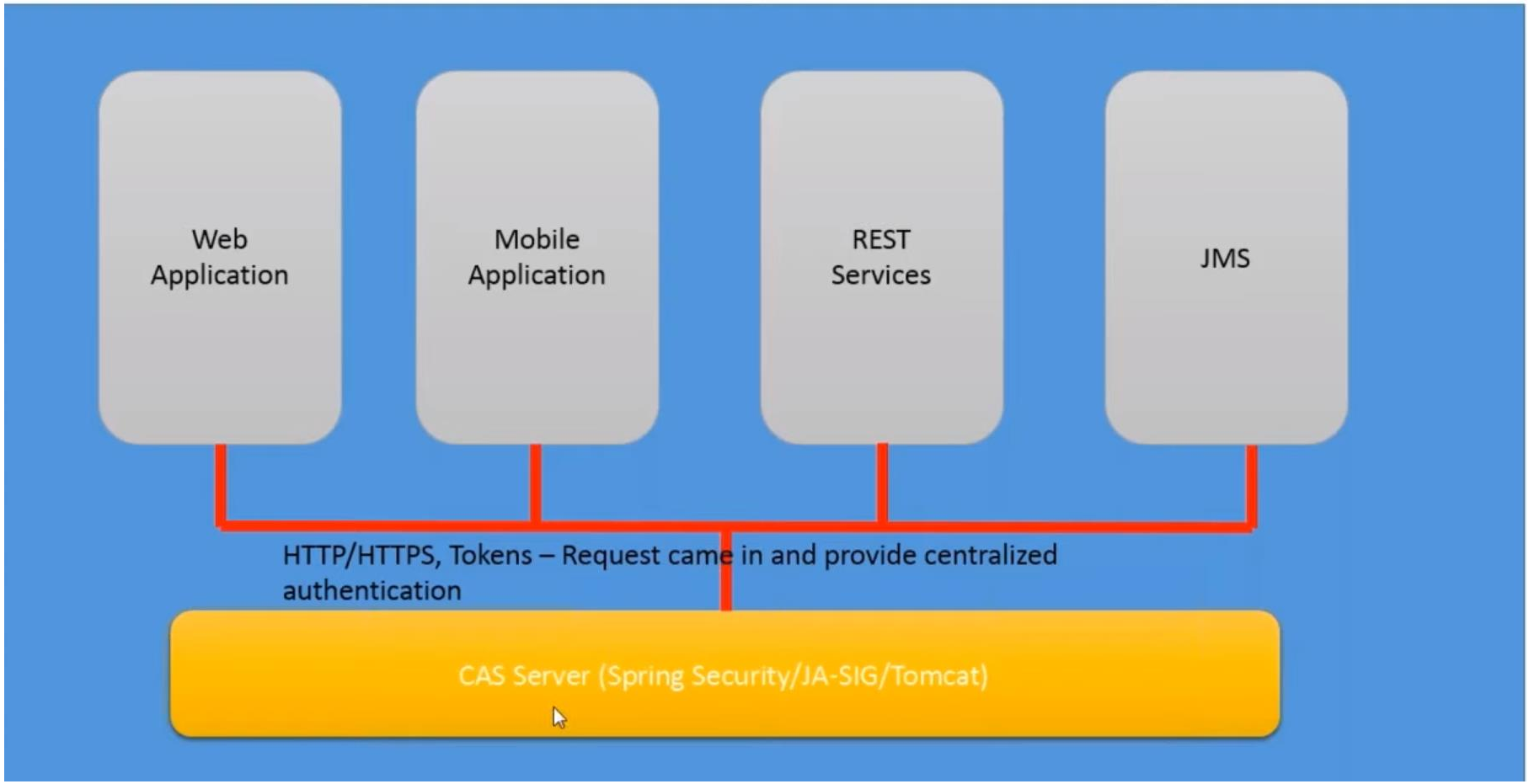
What is Authorization in Spring Security



Authorization

Authentication vs Authorization

Spring Security Features



Spring Security Features

- HTTP BASIC authentication headers
- HTTP Digest authentication headers
- HTTP X.509 client certificate exchange
- LDAP (Lighweight Directory Access Protocol)
- Form-based authentication
- OpenID authentication
- Automatic remember-me authentication
- Kerberos
- JOSSO (Java Open Source Single Sign-On)
- AppFuse
- AndroMDA
- Mule ESB
- DWR(Direct Web Request)

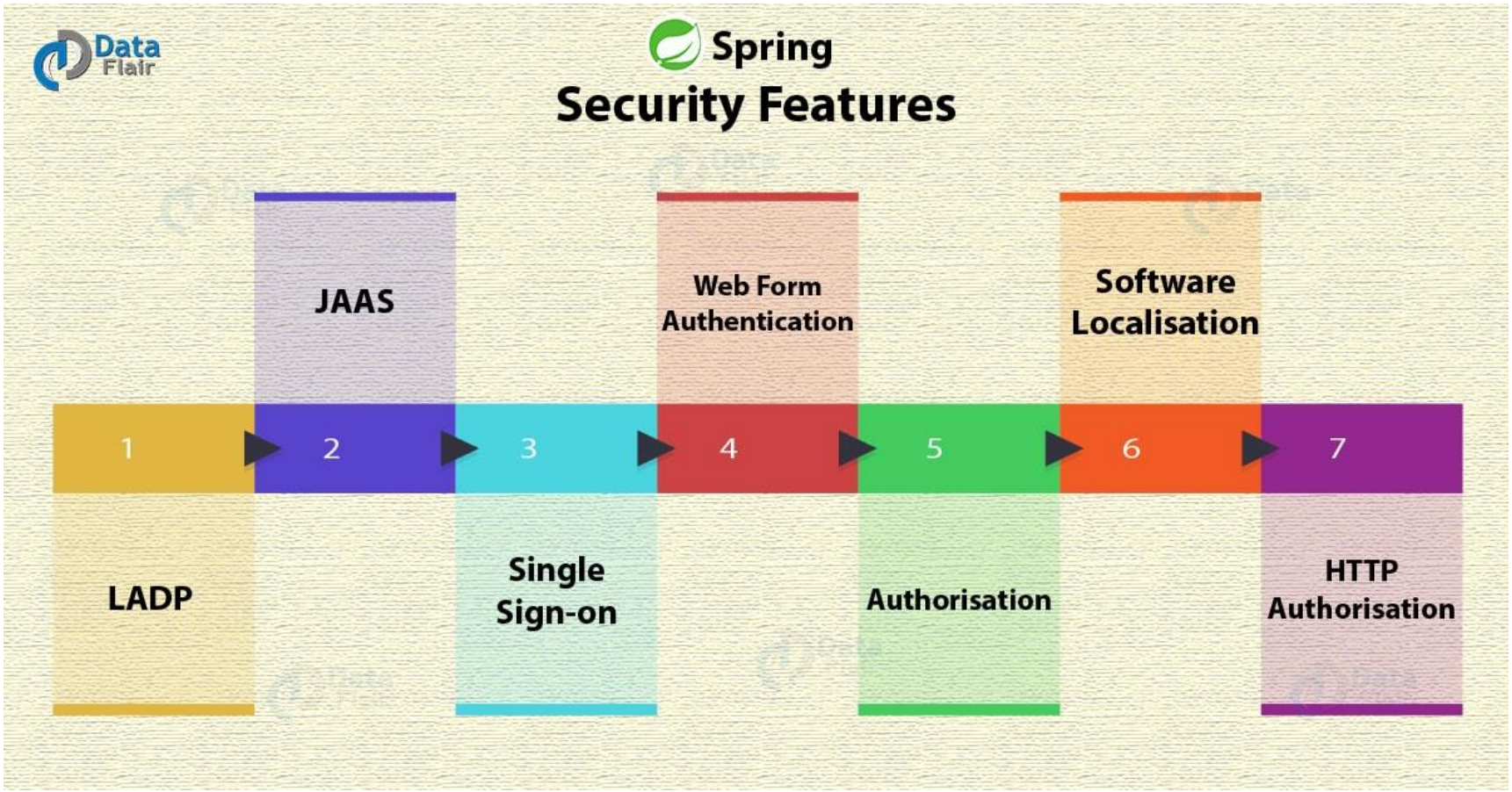
Spring Security Features

- LDAP (Lightweight Directory Access Protocol)
- Single sign-on
- JAAS (Java Authentication and Authorization Service) LoginModule
- Basic Access Authentication
- Digest Access Authentication
- Remember-me

Spring Security Features

- Web Form Authentication
- Authorization
- Software Localization
- HTTP Authorization

Spring Security Features



Spring Security Features

- LDAP (Lightweight Directory Access Protocol)
  - It is an open application protocol for maintaining and accessing distributed directory information services over an Internet Protocol.
- Single sign-on
  - This feature allows a user to access multiple applications with the help of single account(user name and password).
- JAAS (Java Authentication and Authorization Service) LoginModule
  - It is a Pluggable Authentication Module implemented in Java. Spring Security supports it for its authentication process.

Spring Security Features

- Basic Access Authentication
  - Spring Security supports Basic Access Authentication that is used to provide user name and password while making request over the network.
- Digest Access Authentication
  - This feature allows us to make authentication process more secure than Basic Access Authentication. It asks to the browser to confirm the identity of the user before sending sensitive data over the network.
- Remember-me
  - Spring Security supports this feature with the help of HTTP Cookies. It remember to the user and avoid login again from the same machine until the user logout.

Spring Security Features

- Web Form Authentication
  - In this process, web form collect and authenticate user credentials from the web browser. Spring Security supports it while we want to implement web form authentication.
- Authorization
  - Spring Security provides the this feature to authorize the user before accessing resources. It allows developers to define access policies against the resources.
- Software Localization
  - This feature allows us to make application user interface in any language.

Spring Security Features

- HTTP Authorization
  - Spring provides this feature for HTTP authorization of web request URLs using Apache Ant paths or regular expressions.

Features added in Spring Security 5.0

- **OAuth 2.0 Login**
  - This feature provides the facility to the user to login into the application by using their existing account at GitHub or Google.
  - This feature is implemented by using the Authorization Code Grant that is specified in the OAuth 2.0 Authorization Framework.
- **Reactive Support**
  - From version Spring Security 5.0, it provides reactive programming and reactive web runtime support and even, we can integrate with Spring WebFlux.

Features added in Spring Security 5.0

- Modernized Password Encoding
  - Spring Security 5.0 introduced new Password encoder DelegatingPasswordEncoder which is more modernize and solve all the problems of previous encoder NoOpPasswordEncoder.

Spring Security Terms

- Principal
  - User that performs the action
- Authentication
  - Confirming truth of credentials
- Authorization
  - Define access policy for principal
- GrantedAuthority
  - Application permission granted to a principal
- SecurityContext
  - Hold the authentication and other security information
- SecurityContextHolder
  - Provides access to SecurityContext
- AuthenticationManager
  - Controller in the authentication process
- AuthenticationProvider
  - Interface that maps to a data store which stores your user data.
- Authentication Object
  - Object is created upon authentication, which holds the login credentials.
- UserDetails
  - Data object which contains the user credentials, but also the Roles of the user.
- UserDetailsService
  - Collects the user credentials, authorities(roles) and build an UserDetails object.

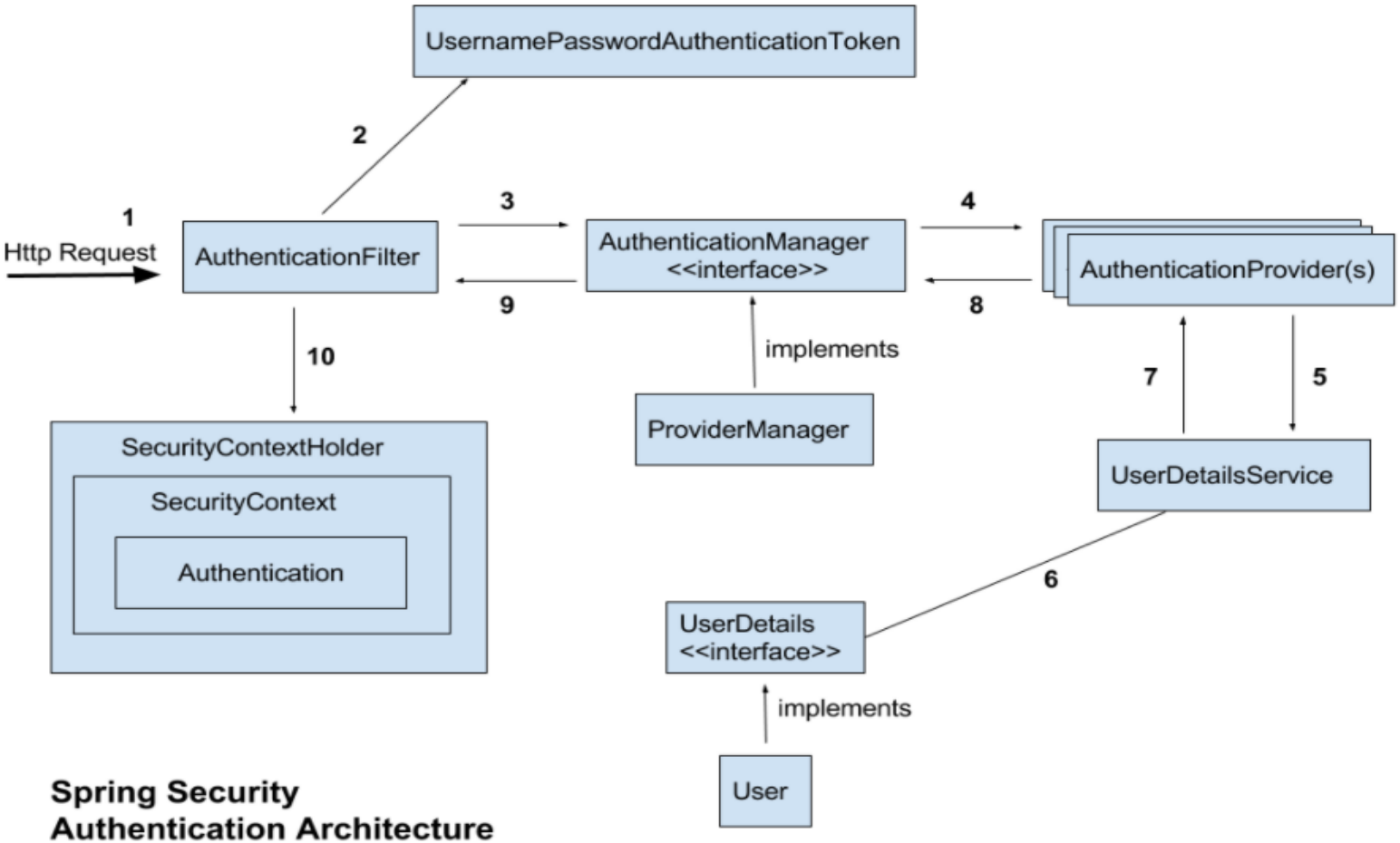
Spring Project Modules

- spring-security-core.jar
- spring-security-remoting.jar
- spring-security-web.jar
- spring-security-config.jar
- spring-security-ldap.jar
- spring-security-oauth2-core.jar
- spring-security-oauth2-client.jar
- spring-security-oauth2-jose.jar
- spring-security-acl.jar
- spring-security-cas.jar
- spring-security-openid.jar
- spring-security-test.jar

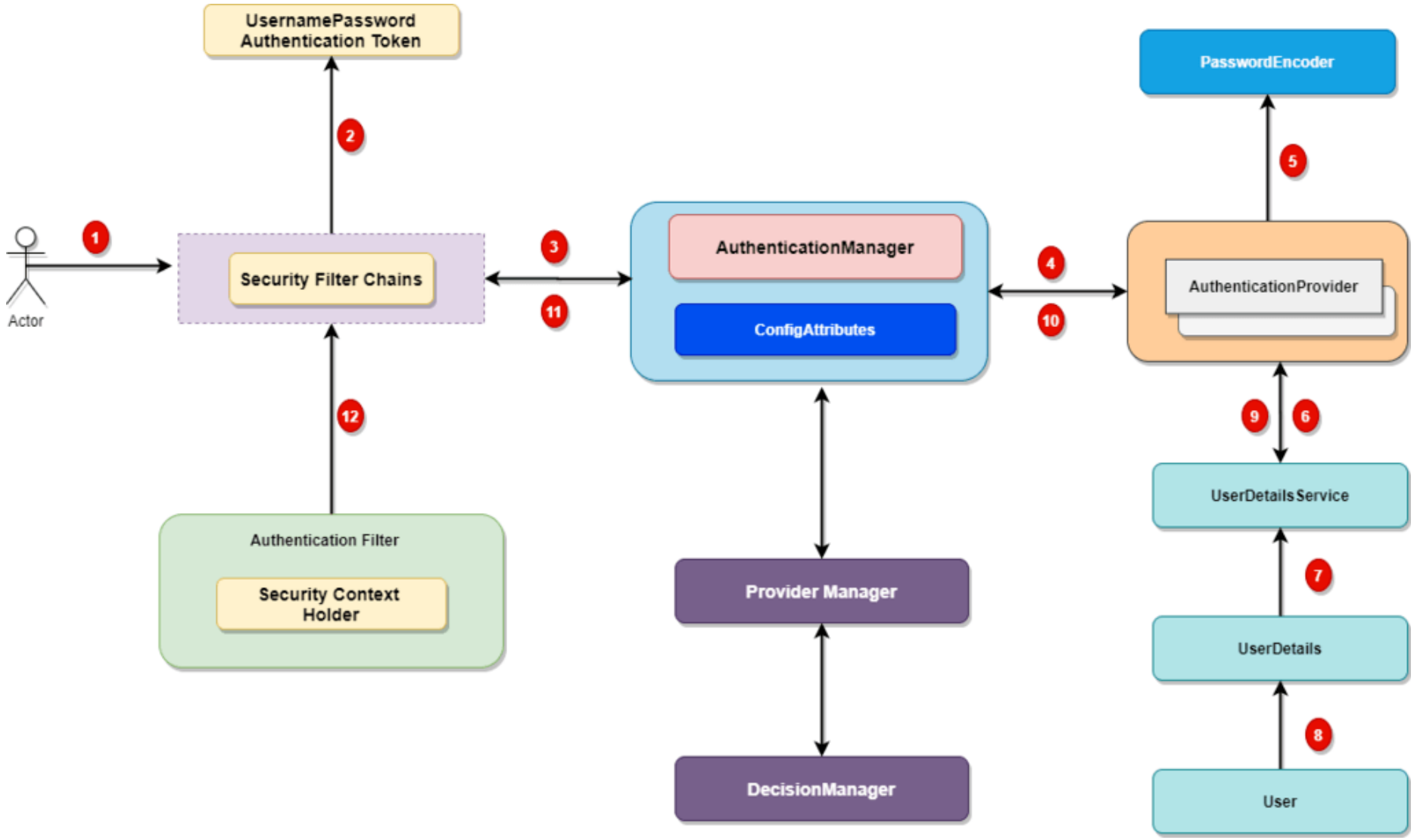
Advantages of Spring Security

- Configuration support to Java Programming Language.
- Portable.
- Comprehensive support to tasks like authorization and authentication.
- Servlet API integration.
- Spring MVC integration.
- CSRF protection.
- Protection against some common tasks.

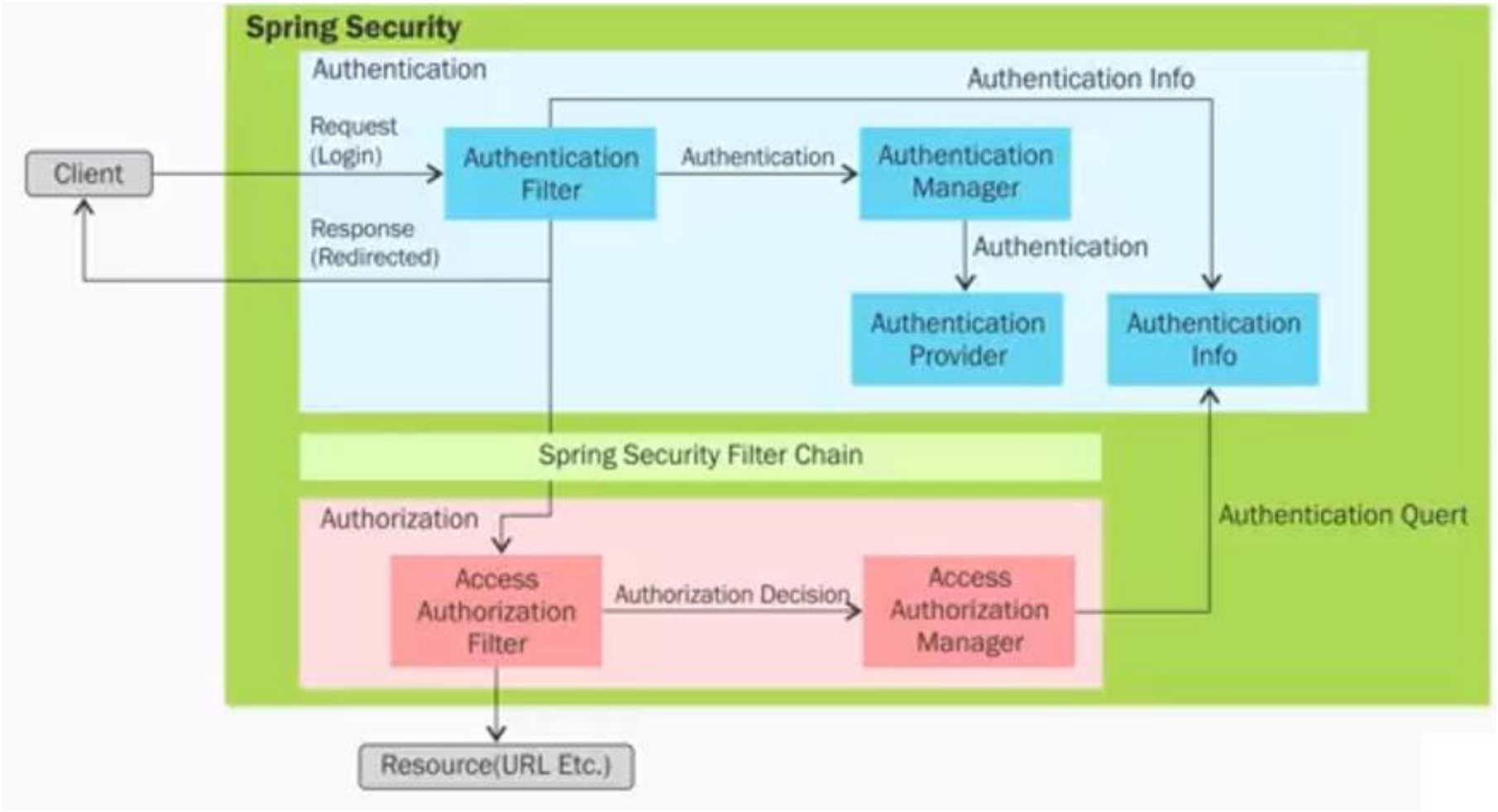
Spring Security Authentication Architecture



Spring Security Authentication Architecture

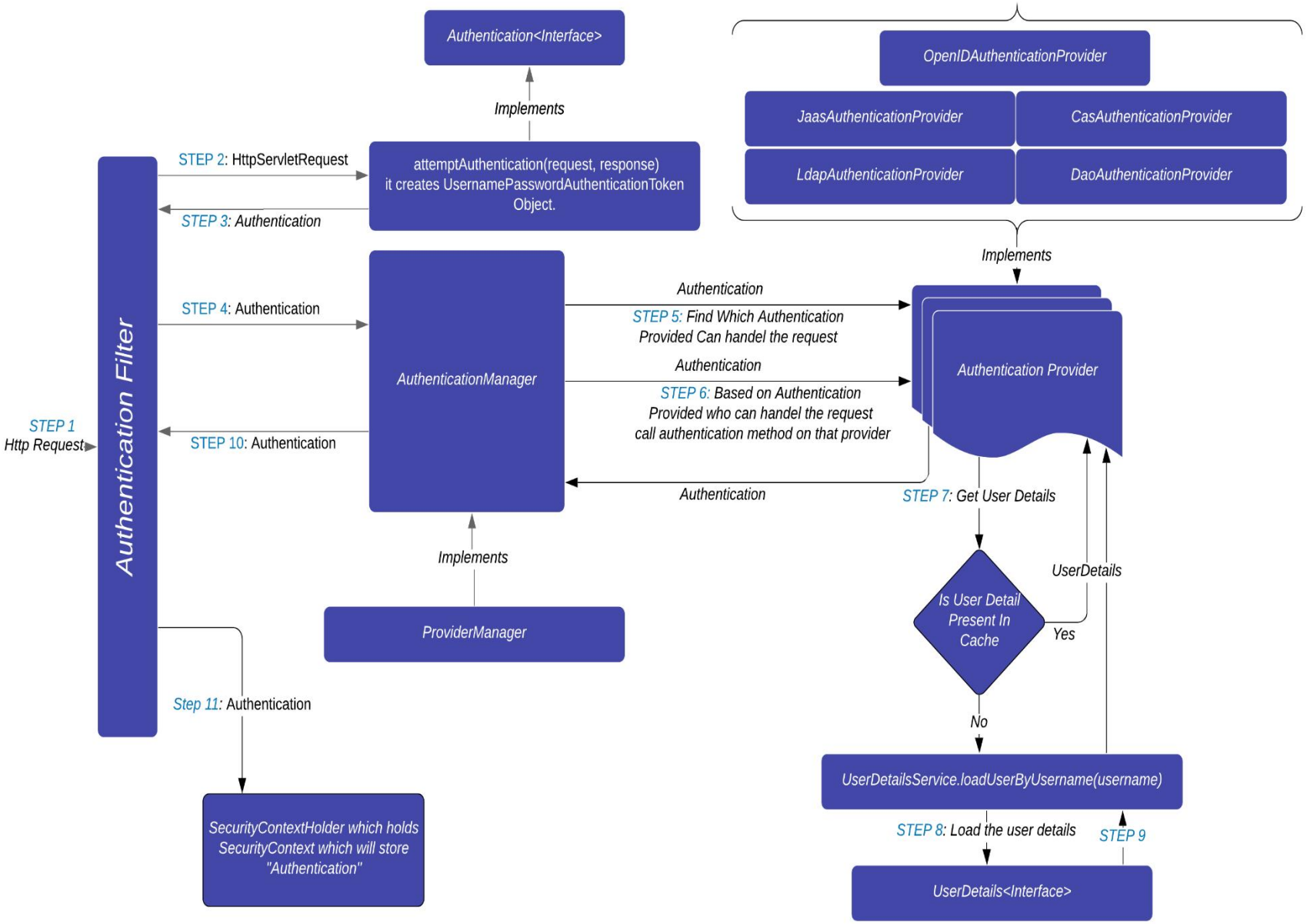


Spring Security Authentication Architecture

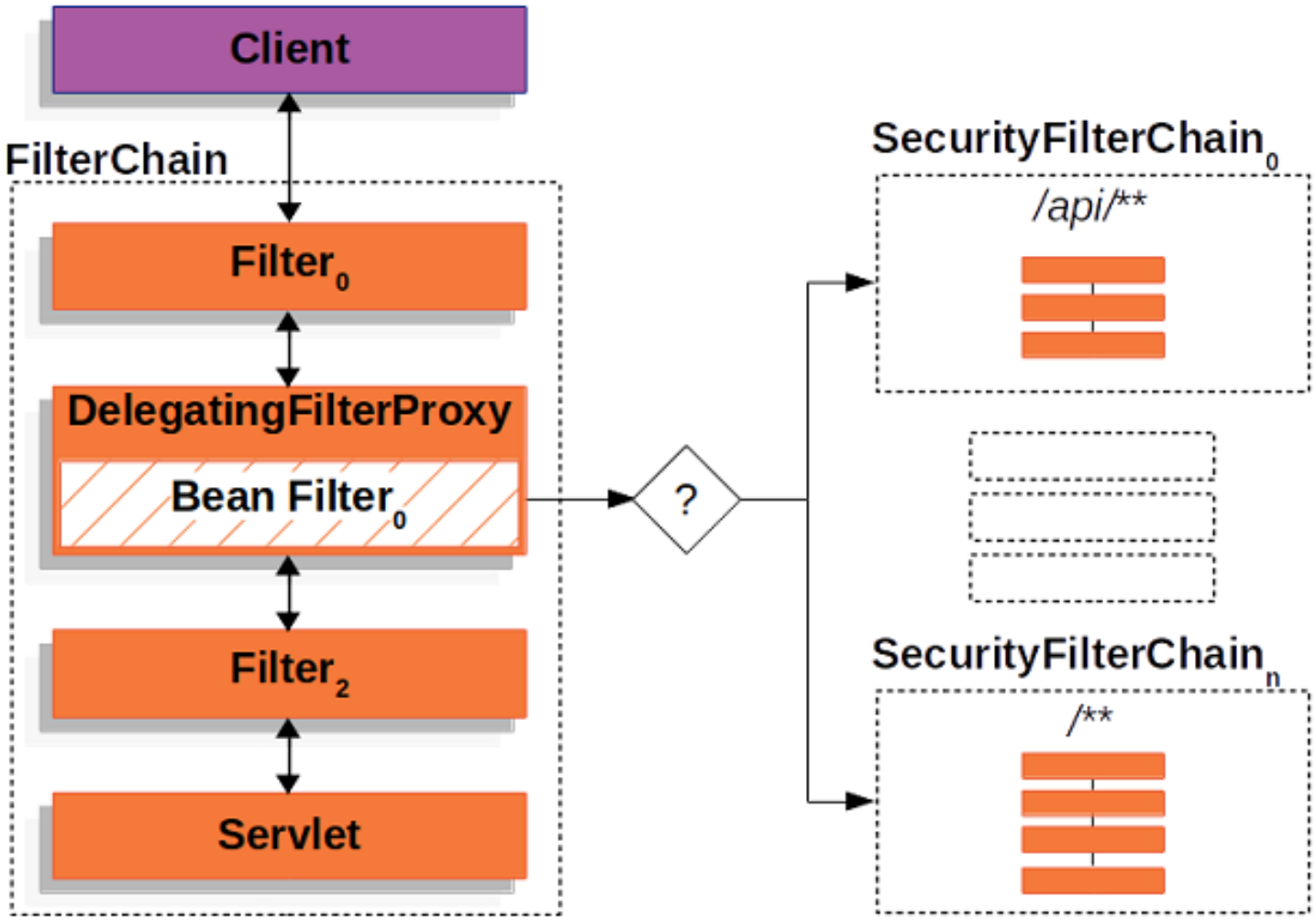


Deep Dive Into Spring Security

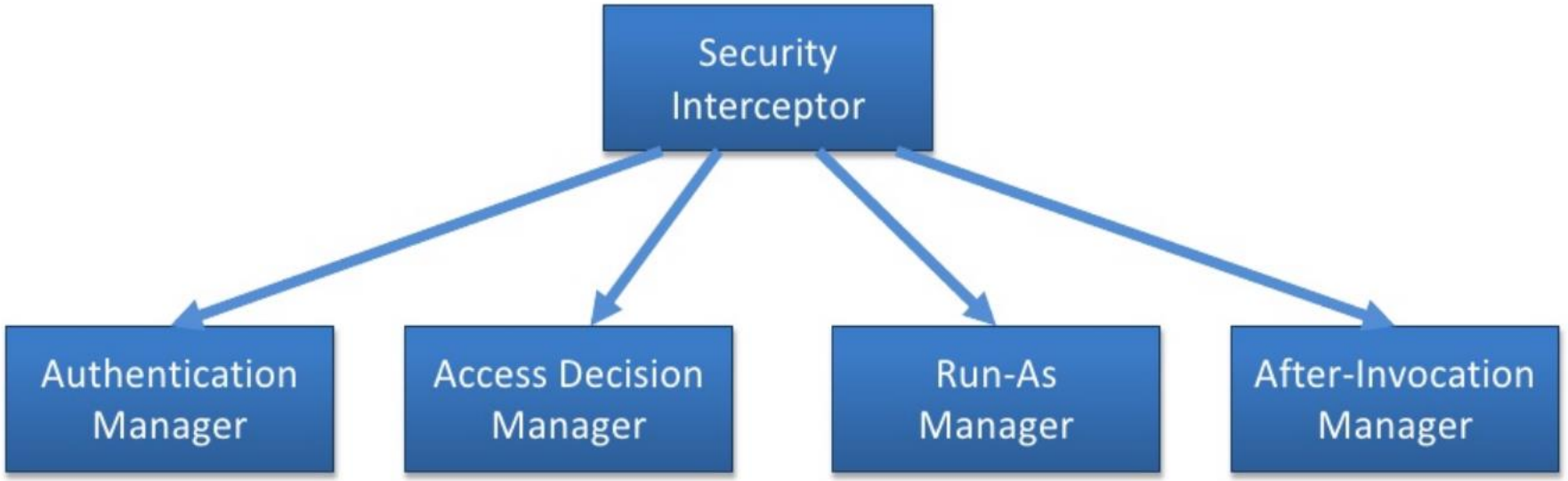
satyakam mohapatra | December 27, 2019



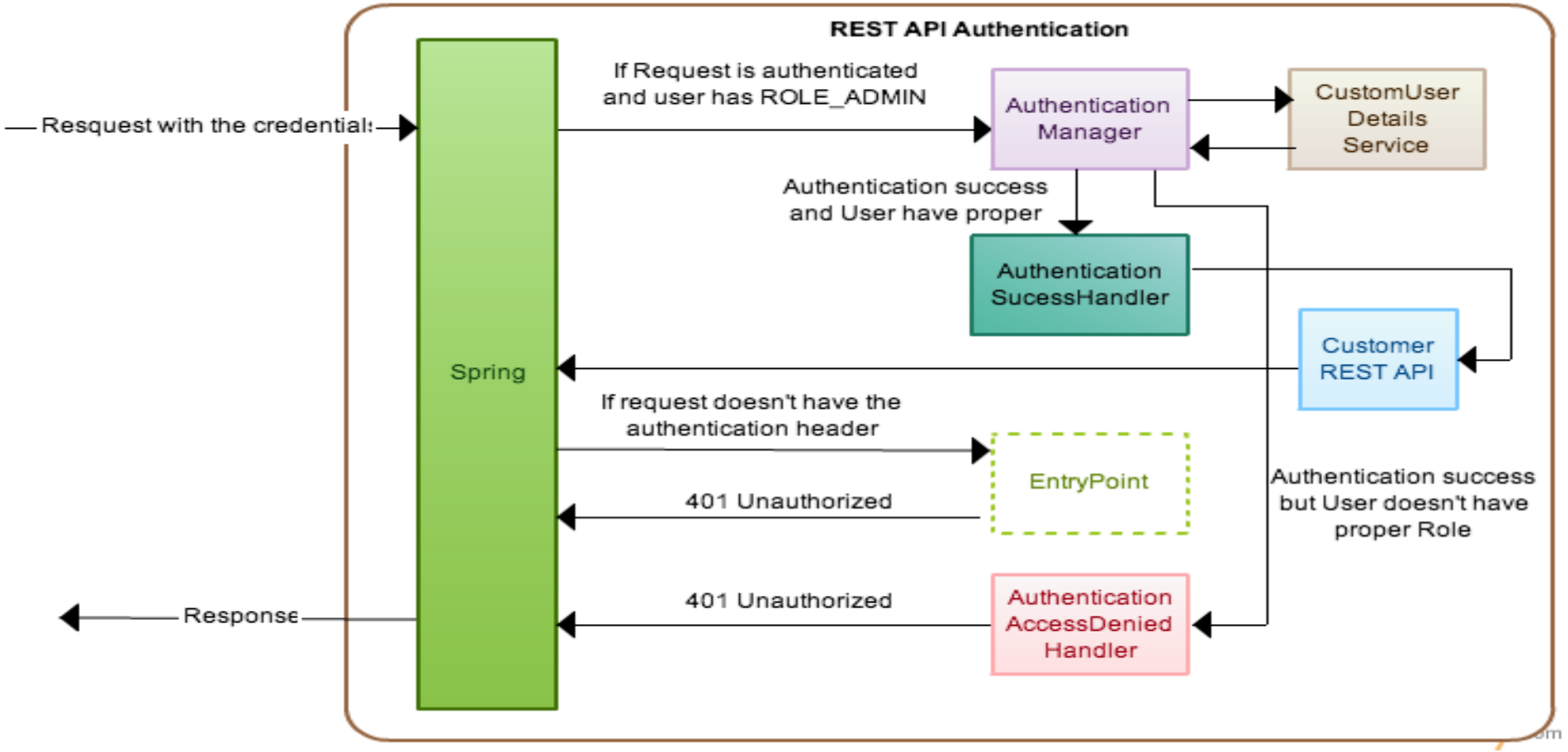
Security Filter Chain



Spring Security Interceptors

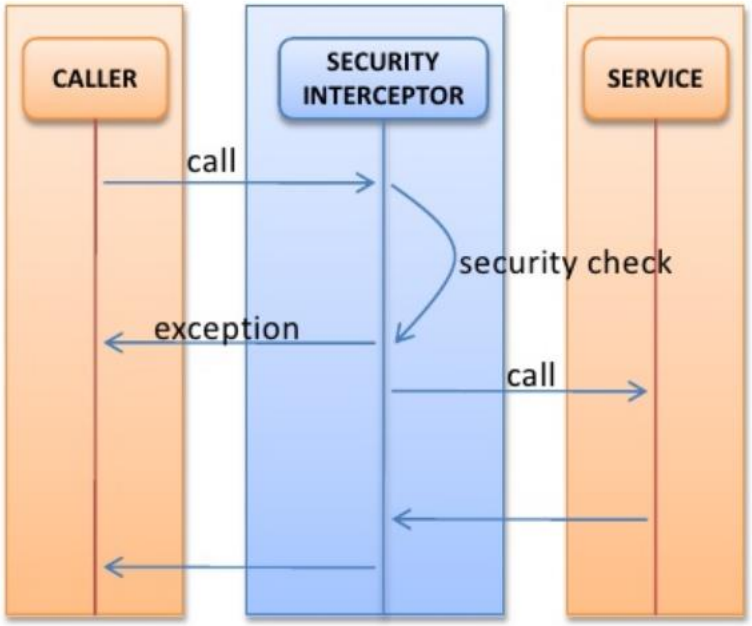


Spring Security



Spring Security Filters

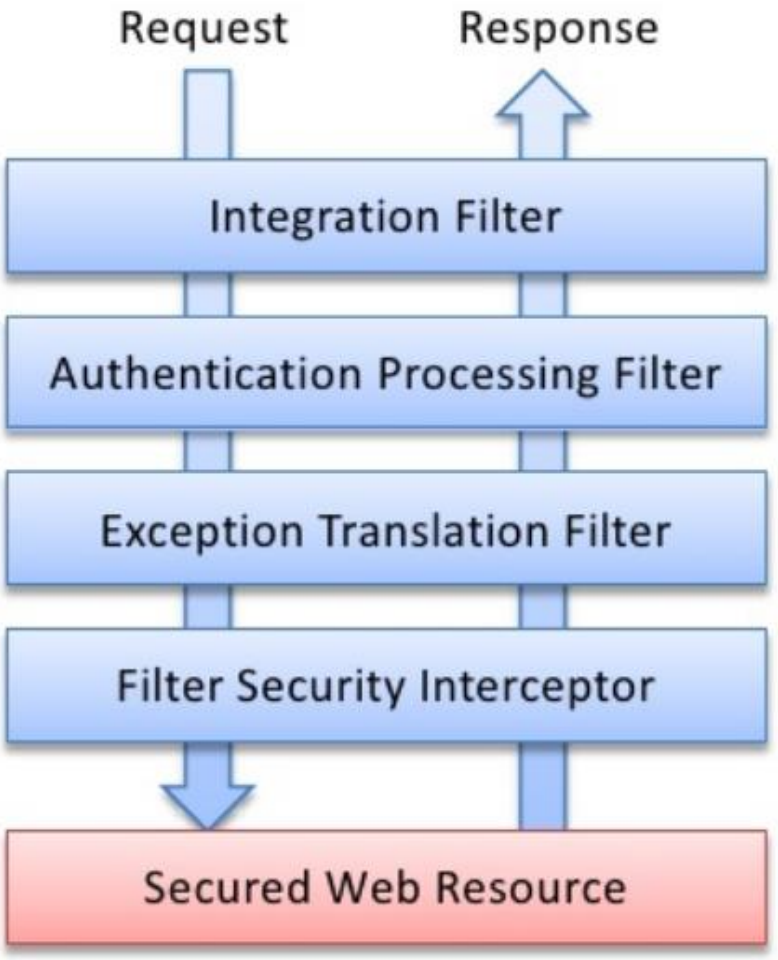
- a latch that protects secured resources, to get past users typically enter a username and password



- implementation depends on resource being secured
  - URLs - Servlet Filter
  - Methods - Aspects

- delegates the responsibilities to the various managers

Spring Security Filters



Spring Security Filters

Filter	What it does
Integration fileter	responsible for retrieving a previously stored authentication (most likely stored in the HTTP session) so that it will be ready for Spring Security's other filters to process.
Authentication Processing Filter	determine if the request is an authentication request. If so, the user information (typically a username / password pair) is retrieved from the request and passed on to the authentication manager.
Exception Translation Filter	translates exceptions, for AuthenticationException request will be sent to a login screen, for AccessDeniedException returns HTTP 403 to the browser.
Filter Security Interceptor	examine the request and determine whether the user has the necessary privileges to access the secured resource. It leans heavily on the authentication manager and the access decision manager.

Spring Security Filters

HttpRequestIntegrationFilter	populates the security context using information from the user principla.
CaptchaValidationProcessingFilter	helps to identify a user as a human using Captcha techniques.
ConcurrentSessionFilter	ensures that a user is not simultaneously logged in more than a set number of times.
HttpSessionContextIntegrationFilter	populates the security context using information obtained from the http session.
FilterSecurityInterceptor	Decides whether or not to allow access to a secured resource.
AnonymousProcessingFilter	Used to identify an unauthenticated user as an anonymous user.
ChannelProcessingFilter	Ensures that a request is being sent over HTTP or HTTPS.

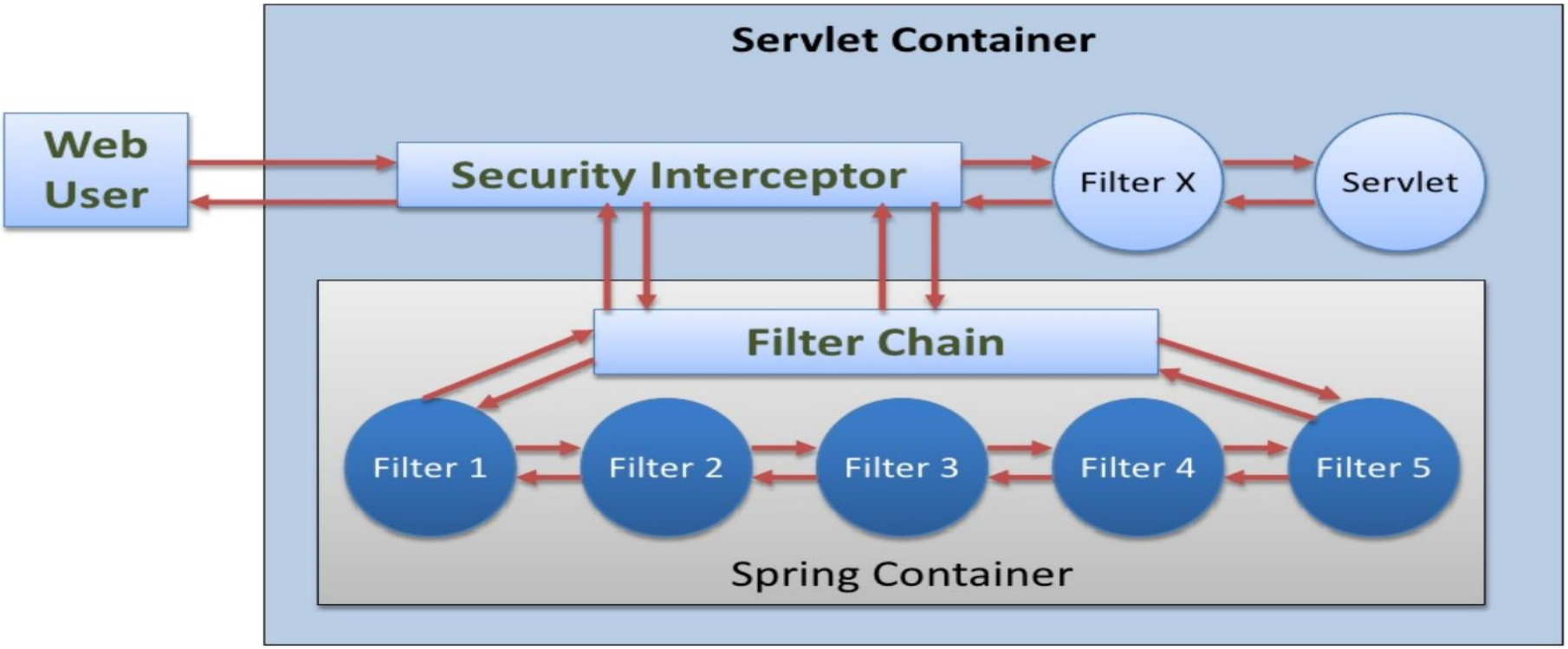
Spring Security Filters

BasicProcessingFilter	Attempts to authenticate a user by processing an HTTP basic authentication.
CasProcessingFilter	Authenticates a user by processing a CAS (Central Authentication Service) ticket.
DigestProcessingFilter	Attempts to authenticate a user by processing an HTTP Digest authentication.
ExceptionTranslationFilter	Handles any AccessDeniedException or AuthenticationException.
LogoutFilter	Used to log a user out of the application.
RememberMeProcessingFilter	Automatically authenticates a user who has asked to be “remembered” by the application.
SwitchUserProcessingFilter	Used to switch out a user. Provides functionality similar to Unix’s su.

Spring Security Filters

AuthenticationProcessingFilter	Accepts the user's principal and credentials and attempts to authenticate the user.
SiteminderAuthenticationProcessingFilter	Authenticates a user by processing CA/Netegrity SiteMinder headers.
X509ProcessingFilter	Authenticates a user by processing an X.509 certificate submitted by a client web browser.
SecurityContextHolderAwareRequestFilter	populates the servlet request with a request wrapper.

Flow of a request through Spring Security's core filters

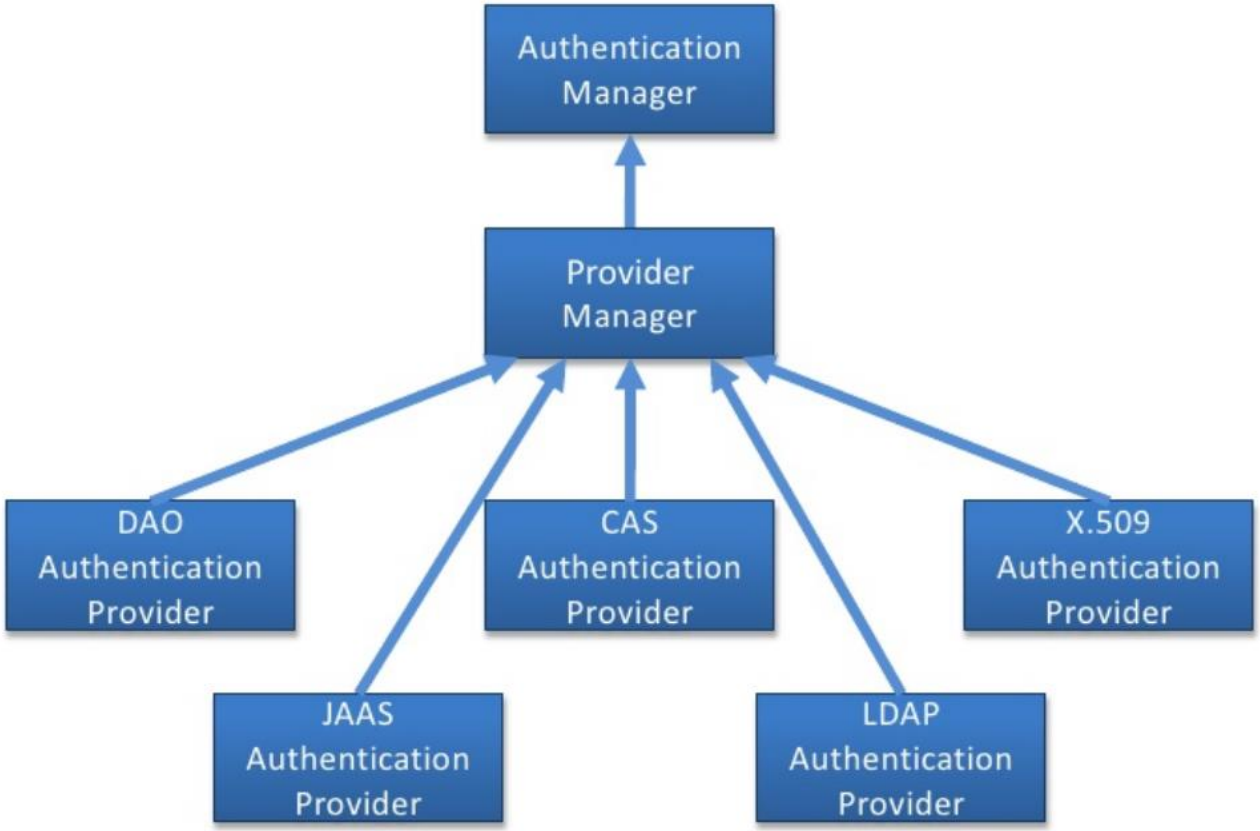


Authentication

Authentication Manager

- verifies *principal (typically a username) and credentials* (typically a password)

- Spring Security comes with a handful of flexible authentication managers that cover the most common authentication strategies



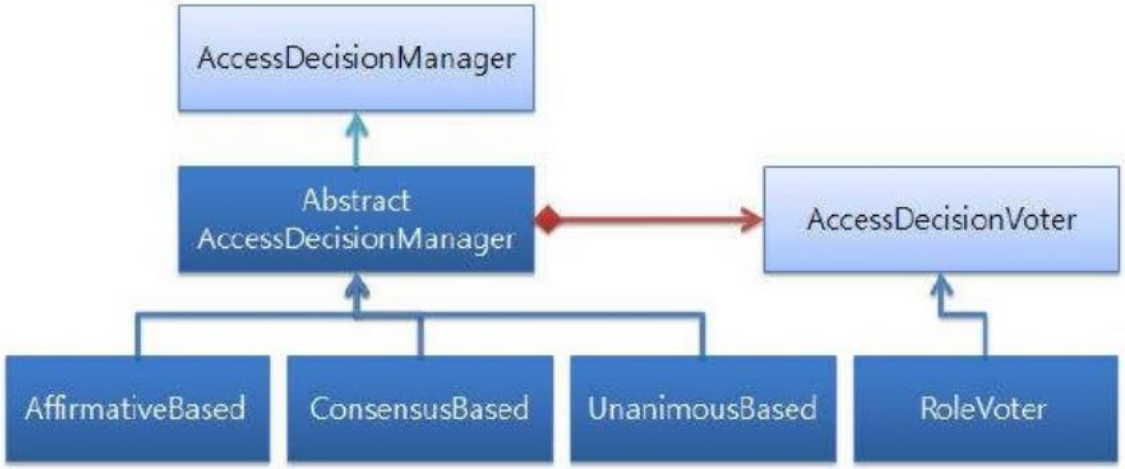
Spring Security

Authorization

Access Decision Manager

- responsible for deciding whether the user has the proper access to secured resources by invoking Voters and tallying votes

- Spring Security comes with three implementations of Access Decision Manager



Access decision manager	How it decides to grant/deny access
Affirmative Based	Allows access if at least one voter votes to grant access
Consensus Based	Allows access if a consensus of voters vote to grant access
Unanimous Based	Allows access if all voters vote to grant access

Creating a Keystore File and Keystore Password for HTTPS Connections

- % keytool -genkey -alias tomcat -keyalg RSA -keystore /keystore-location -storepass password

- keytool -genkey -noprompt -alias <your-alias> -keyalg RSA -keystore <your-file-name> -keypass <your-password>
- -storepass <your-password> -dname "CN=<your-cert-name>, OU=<your-organization-unit>, O=<your-organization>,
- L=<your-location>, ST=<state>, C=<two-letter-country-code>"

Creating a Keystore File and Keystore Password for HTTPS Connections

- **keytool -genkey -noprompt -alias tomcat-localhost -keyalg RSA -keystore localhost-rsa.jks -keypass test@123 -storepass test@123 -dname "CN=tomcat-cert, OU=Dev, O=Virtusa, L=Chennai, ST=TN, C=IN"**
- **Enhance the certificate**
- **keytool -importkeystore -srckeystore localhost-rsa.jks -destkeystore banking-rsa.jks -deststoretype pkcs12 -noprompt -alias tomcat-localhost -keyalg RSA -keypass test@123 -storepass test@123 -dname "CN=tomcat-cert, OU=Dev, O=Virtusa, L=Chennai, ST=TN, C=IN"**

How to Obtain a Signature for an SSL Certificate

- **Generate**
- **keytool -certreq -v -alias tomcat -keyalg RSA -keystore /keystore-location**
- **keytool -certreq -v -alias tomcat-localhost -keyalg RSA -keystore banking-rsa.jks**
- **Print certificate**
- **% keytool -printcert -file certificate-reply-file**
- **keytool -v -import -trustcacerts -keystore /keystore-location -file certificate-reply-file -alias tomcat**
- **keytool -v -list -keystore /keystore-location**

How to configure Maven Tomcat Plugin to use HTTPS (SSL/TLS)

(1)Creating a Keystore

(2)Configuring tomcat7-maven-plugin to use HTTPS

```
<plugin>
    <groupId>org.apache.tomcat.maven</groupId>
    <artifactId>tomcat7-maven-plugin</artifactId>
    <version>2.2</version>
    <configuration>
        <path>/</path>
        <httpsPort>8443</httpsPort>
        <keystoreFile>C:\my-cert-dir\localhost-rsa.jks</keystoreFile>
        <keystorePass>123456</keystorePass>
    </configuration>
 </plugin>
```

User credentials are sent in clear / easy to decode format

- Password Hashing
- Hash algorithms are one way functions. They turn any amount of data into a fixed-length "fingerprint" that cannot be reversed.
- They also have the property that if the input changes by even a tiny bit, the resulting hash is completely different.
- This is great for protecting passwords, because we want to store passwords in a form that protects them even if the password file itself is compromised.
- But at the same time, we need to be able to verify that a user's password is correct.

User credentials are sent in clear / easy to decode format

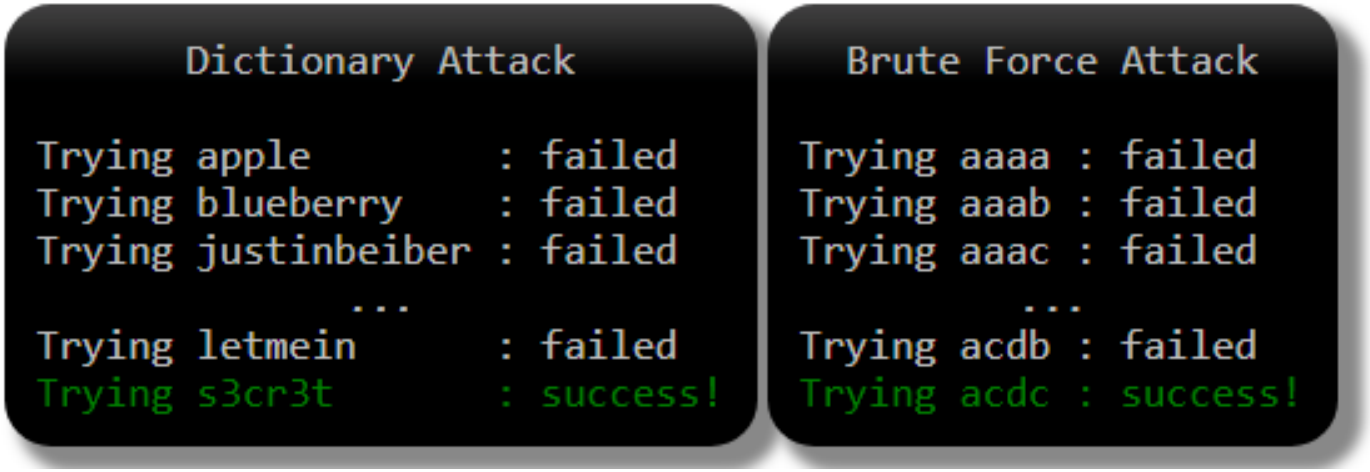
- The general workflow for account registration and authentication in a hash-based account system is as follows:
- The user creates an account.
  - Their password is hashed and stored in the database. At no point is the plain-text (unencrypted) password ever written to the hard drive.
  - When the user attempts to login, the hash of the password they entered is checked against the hash of their real password (retrieved from the database).
  - If the hashes match, the user is granted access. If not, the user is told they entered invalid login credentials.
  - Steps 3 and 4 repeat every time someone tries to login to their account.
  - In step 4, never tell the user if it was the username or password they got wrong. Always display a generic message like "Invalid username or password." This prevents attackers from enumerating valid usernames without knowing their passwords.

User credentials are sent in clear / easy to decode format

- The hash functions used to implement data structures such as hash tables are designed to be fast, not secure.
- Only cryptographic hash functions may be used to implement password hashing.
- Hash functions like SHA256, SHA512, RipeMD, and WHIRLPOOL are cryptographic hash functions.

User credentials are sent in clear / easy to decode format

- How Hashes are Cracked
- Dictionary and Brute Force Attacks



User credentials are sent in clear / easy to decode format

- How Hashes are Cracked
- **Lookup Tables**
  - Lookup tables are an extremely effective method for cracking many hashes of the same type very quickly.
  - The general idea is to pre-compute the hashes of the passwords in a password dictionary and store them, and their corresponding password, in a lookup table data structure.
  - A good implementation of a lookup table can process hundreds of hash lookups per second, even when they contain many billions of hashes.

User credentials are sent in clear / easy to decode format

- How Hashes are Cracked
- **Lookup Tables**

```
Searching: 5f4dcc3b5aa765d61d8327deb882cf99: FOUND: password5
Searching: 6cbe615c106f422d23669b610b564800:  not in database
Searching: 630bf032efe4507f2c57b280995925a9: FOUND: letMEin12
Searching: 386f43fab5d096a7a66d67c8f213e5ec: FOUND: mcd0nalds
Searching: d5ec75d5fe70d428685510fae36492d9: FOUND: p@ssw0rd!
```

User credentials are sent in clear / easy to decode format

- How Hashes are Cracked
- The WRONG Way: Short Salt & Salt Reuse
- Salt Reuse
- A common mistake is to use the same salt in each hash.
- Either the salt is hard-coded into the program, or is generated randomly once.
- This is ineffective because if two users have the same password, they'll still have the same hash.
- An attacker can still use a reverse lookup table attack to run a dictionary attack on every hash at the same time.
- They just have to apply the salt to each password guess before they hash it.
- If the salt is hard-coded into a popular product, lookup tables and rainbow tables can be built for that salt, to make it easier to crack hashes generated by the product.

User credentials are sent in clear / easy to decode format

- Short Salt
- If the salt is too short, an attacker can build a lookup table for every possible salt. For example, if the salt is only three ASCII characters, there are only 95x95x95 = 857,375 possible salts.
- That may seem like a lot, but if each lookup table contains only 1MB of the most common passwords, collectively they will be only 837GB, which is not a lot considering 1000GB hard drives can be bought for under $100 today.
- For the same reason, the username shouldn't be used as a salt.
- Usernames may be unique to a single service, but they are predictable and often reused for accounts on other services.
- An attacker can build lookup tables for common usernames and use them to crack username-salted hashes.
- To make it impossible for an attacker to create a lookup table for every possible salt, the salt must be long.
- **A good rule of thumb is to use a salt that is the same size as the output of the hash function. For example, the output of SHA256 is 256 bits (32 bytes), so the salt should be at least 32 random bytes.**

User credentials are sent in clear / easy to decode format

- The WRONG Way: Double Hashing & Wacky Hash Functions
- Here are some examples of poor wacky hash functions I've seen suggested in forums on the internet.
  - md5(sha1(password))
  - md5(md5(salt) + md5(password))
  - sha1(sha1(password))
  - sha1(str_rot13(password + salt))
  - md5(sha1(md5(md5(password) + sha1(password)) + md5(password)))
  - **Do not use any of these.**

User credentials are sent in clear / easy to decode format

- The RIGHT Way: How to Hash Properly
- Salt should be generated using a Cryptographically Secure Pseudo-Random Number Generator (CSPRNG).
- CSPRNGs are very different than ordinary pseudo-random number generators, like the "C" language's rand() function.
- As the name suggests, CSPRNGs are designed to be cryptographically secure, meaning they provide a high level of randomness and are completely unpredictable.
- We don't want our salts to be predictable, so we must use a CSPRNG.

User credentials are sent in clear / easy to decode format

- java.lang.Object
- extended by java.util.Random
- extended by java.security.SecureRandom
- SecureRandom random = new SecureRandom();
- byte bytes[] = new byte[20];
- random.nextBytes(bytes);

User credentials are sent in clear / easy to decode format

- To Store a Password
  - Generate a long random salt using a CSPRNG.
  - Prepend the salt to the password and hash it with a standard password hashing function like Argon2, bcrypt, scrypt, or PBKDF2.
  - Save both the salt and the hash in the user's database record.
- To Validate a Password
  - Retrieve the user's salt and hash from the database.
  - Prepend the salt to the given password and hash it using the same hash function.
  - Compare the hash of the given password with the hash from the database. If they match, the password is correct. Otherwise, the password is incorrect.

User credentials are sent in clear / easy to decode format

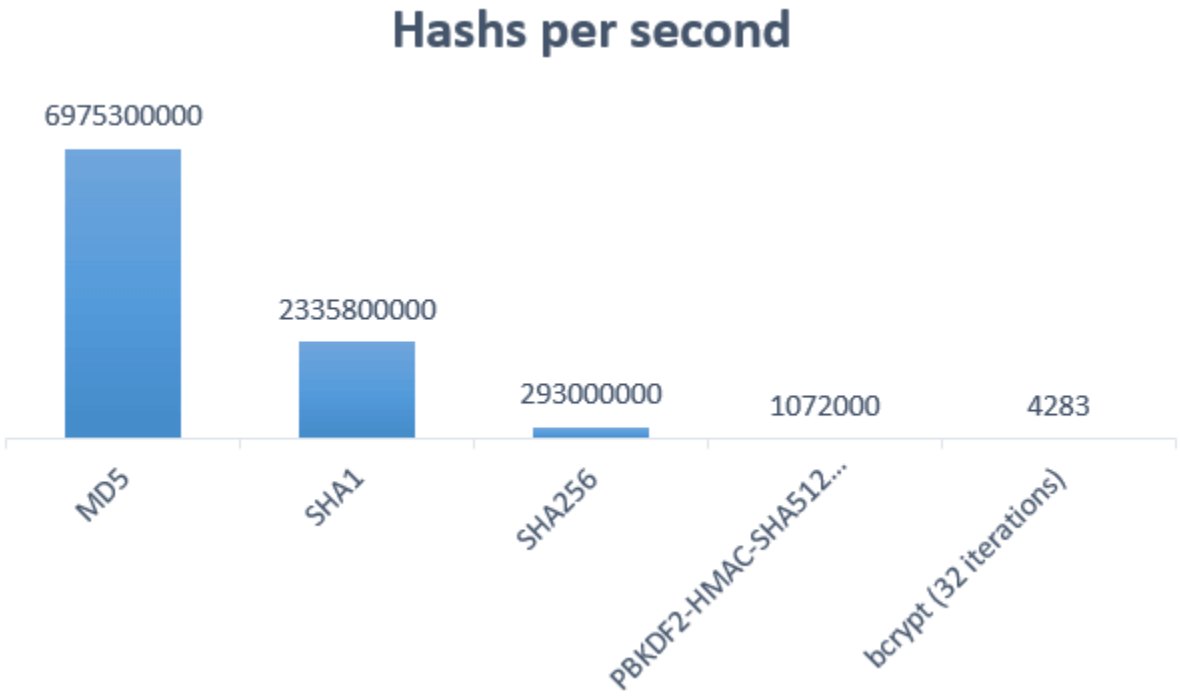
- Making Password Cracking Harder: Slow Hash Functions
- Use a standard algorithm like PBKDF2 or bcrypt.

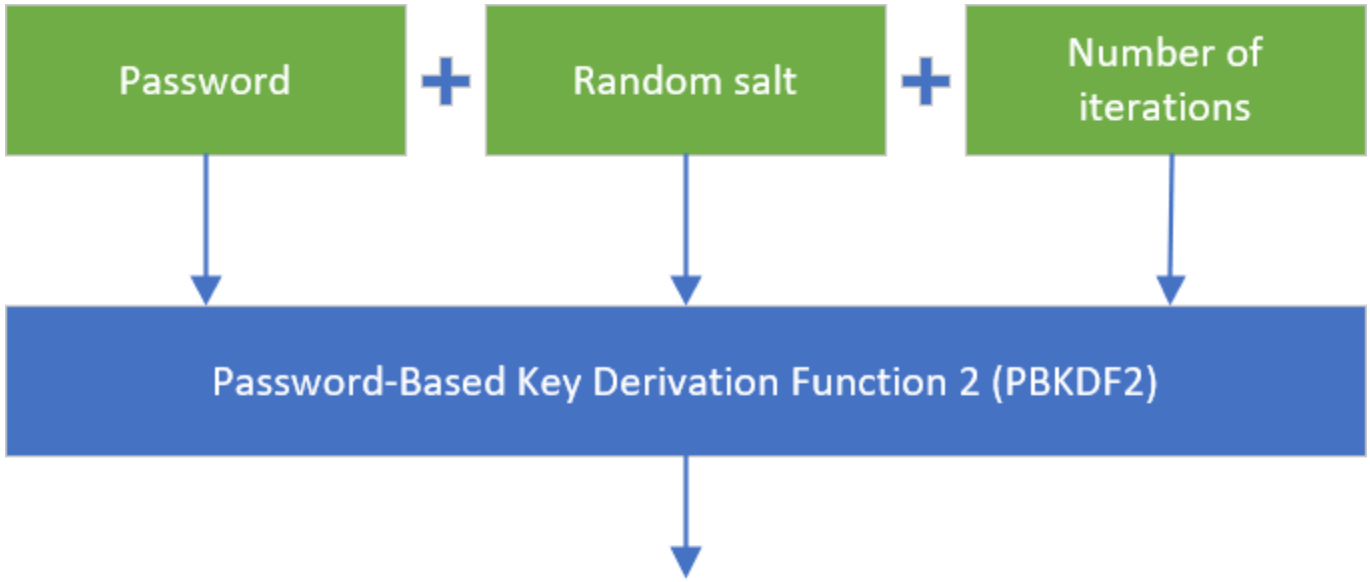
User credentials are sent in clear / easy to decode format

- Hashing Passwords in Java with Bcrypt.
- BCrypt is a one way salted hash function based on the Blowfish cipher.
- It provides several enhancements over plain text passwords.

User credentials are sent in clear / easy to decode format

- BCrypt features
- **Not plain text** - Not only do plain text passwords compromise your website if the database is breached but they can also compromise other websites for the users. Unfortunately a lot of users share passwords across websites.
- **One way hashing** - BCrypt is a one way hash function to obfuscate the password such that it is not stored in plain text.
- **Salted hashing** - Generating random bytes (the salt) and combining it with the password before hashing creates unique hashes across each users password. If two users have the same password they will not have the same password hash. This is to prevent rainbow table attacks which can reverse hashed passwords using common hashing functions that do not utilize a salt.
- **Logarithmic iterations** - The hashing function is executed many times sequentially which can be increased exponentially known as key stretching. This is to make the function CPU intensive which makes it more secure against brute force attacks.
- **Updatable iterations** - As CPUs become faster so do brute force attacks. Since BCrypt stores the number of iterations as part of the hash it's possible to verify a password and then increase its strength by generating a new hash with a higher number of iterations.


Hashs per second
6975300000
2335800000
293000000
1072000
4283
MD5
SHA1
SHA256
PBKDF2-HMAC-SHA512...
bcrypt (32 iterations)


Password
Random salt
Number of iterations
Password-Based Key Derivation Function 2 (PBKDF2)
65c23f142ff6bcfdddeccebc0e5e63c41c9c1721

Common Security Vulnerabilities

- Broken authentication
- Session fixation
- Cross-site scripting (XSS)
- Cross-site request forgery (CSRF)
- Injections
- Sensitive data exposure
- Lack of method access control
- Using dependencies with known vulnerabilities

Host Header Attack

- The “HOST” header is part of the http protocol, vulnerable applications are vulnerable because they insert the value of this header into the application code without proper validation.
- It means not only applications hosted on Apache/Nginx can be vulnerable.
- For Host Header Attack Exploitation, basically there are two ways through which you can exploit the application i.e.
- By means of Web Cache Poisoning which manipulates caching systems into storing a page generated with a malicious host .
- other is via Password Reset Emails when poisoned content is delivered directly to the target.

Why HTTP host header needed ?

- According to RFC 2616: A client MUST include a Host header field in all HTTP/1.1 request messages.
- If the requested URI does not include an Internet host name for the service being requested, then the Host header field MUST be given with an empty value.
- Any HTTP/1.1 request without host header field must be responded by server with 400(bad request) status code.

Why HTTP host header needed ?

- There are several different types of attacks related to host header injection
- Unauthorized URL Redirect by Cache poisoning
- Password Reset Poisoning
- Access to internal hosts
- Cross site scripting

Why HTTP host header needed ?

- There are several different types of attacks related to host header injection
- Unauthorized URL Redirect by Cache poisoning
- Password Reset Poisoning
- Access to internal hosts
- Cross site scripting

Prevent Host Header Attack

- <a href="<?=$_SERVER['HTTP_HOST']?>/blog/">Blog

HTML Injection

- The essence of this type of injection attack is injecting HTML code through the vulnerable parts of the website.
- The Malicious user sends HTML code through any vulnerable field with a purpose to change the website's design or any information, that is displayed to the user.
- In the result, the user may see the data, that was sent by the malicious user.
- Therefore, in general, HTML Injection is just the injection of markup language code to the document of the page.
- Data, that is being sent during this type of injection attack may be very different.
- It can be a few HTML tags, that will just display the sent information. Also, it can be the whole fake form or page.
- When this attack occurs, the browser usually interprets malicious user data as legit and displays it.

Types of HTML Injection

- Firstly, different types may be sorted by the risks, that they bring.
- As mentioned, this injection attack can be performed with two different purposes:
  - To change the displayed website’s appearance.
  - To steal another person’s identity.
- However, the main types are:
  - Stored HTML Injection
  - Reflected HTML Injection

Stored HTML Injection:

- Stored injection attack occurs when malicious HTML code is saved in the web server and is being executed every time when the user calls an appropriate functionality.

Reflected HTML Injection:

- This can be again divided into more types:
- Reflected GET
- Reflected POST
- Reflected URL

For Example, a tester can check the source code for the login form and find what method is being used for it. Then appropriate HTML Injection method can be selected accor

```
▼<div class="test">
  ▼<form id="test-login" method="post" action="#">
      <input type="text" id="test-username" class name="user"
      placeholder="Email">
      <input type="password" id="test-password" class name="pass"
```

Reflected HTML Injection:

- **Reflected GET Injection** occurs, when our input is being displayed (reflected) on the website.
- Suppose, we have a simple page with a search form, which is vulnerable to this attack.
- Then if we would type any HTML code, it will appear on our website and at the same time, it will be injected into the HTML document.

How to Prevent HTML Injection?

- There are no doubts, that the main reason for this attack is the developer's inattention and lack of knowledge.
- This type of injection attack occurs when the input and output are not properly validated.
- Therefore the main rule to prevent HTML attack is appropriate data validation.

Authentication Bypass

- Applications require authentication for gaining access to private information.
- Not every authentication method is able to provide adequate security.
- A flaw in the application that allows users to access application resources without authentication is referred as’ Authentication Bypass’.
- Authentication bypass vulnerability is generally caused when it is assumed that users will behave in a certain way and failing to foresee the consequences of users doing the unexpected.

Methods to bypass the authentication schema:

- There are several methods to bypass the authentication schema in use by a web application. Here are some of the common ways to bypass authentication:
- Direct page request (Forced Browsing)
- Parameter Modification
- Session ID Prediction
- SQL Injection

Test Authorization Bypass

- Testing for access to administrative functions
- For example, suppose that the 'AddUser.jsp' function is part of the administrative menu of the application, and it is possible to access it by requesting the following URL:
- https://www.example.com/admin/addUser.jsp

Test Authorization Bypass

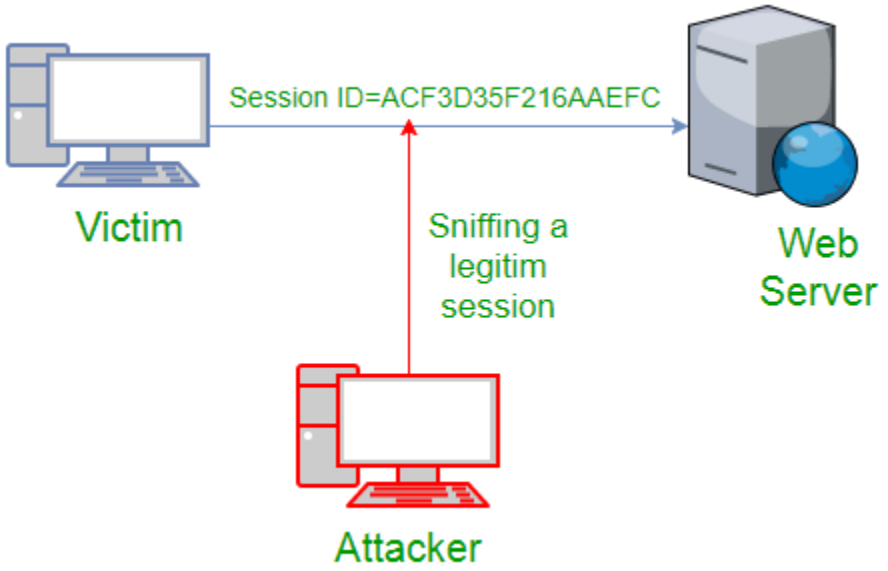
- Then, the following HTTP request is generated when calling the AddUser function:
- POST /admin/addUser.jsp HTTP/1.1
- Host: www.example.com
- [other HTTP headers]
- userID=fakeuser&role=3&group=grp001
- What happens if a non-administrative user tries to execute that request? Will the user be created? If so, can the new user use their privileges?

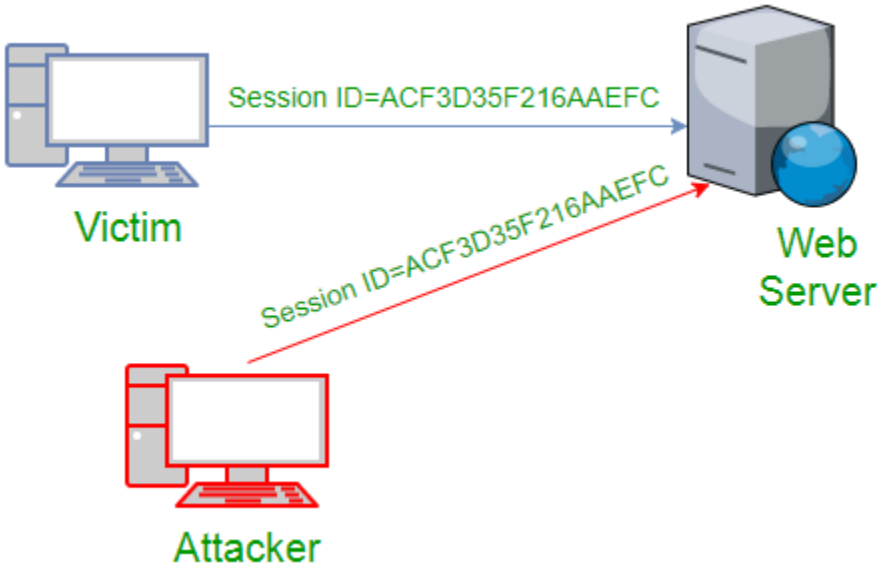
Session Hijacking

- TCP session hijacking is a security attack on a user session over a protected network.
- The most common method of session hijacking is called IP spoofing, when an attacker uses source-routed IP packets to insert commands into an active communication between two nodes on a network and disguising itself as one of the authenticated users.
- This type of attack is possible because authentication typically is only done at the start of a TCP session.

Session Hijacking

- Another type of session hijacking is known as a man-in-the-middle attack, where the attacker, using a sniffer, can observe the communication between devices and collect the data that is transmitted.


Session ID=ACF3D35F216AAEFC
Victim
Sniffing a legitim session
Web Server
Attacker


Session ID=ACF3D35F216AAEFC
Victim
Session ID=ACF3D35F216AAEFC
Web Server
Attacker

Session Hijacking

- Cross Site Scripting(XSS Attack)
- Attacker can also capture victim’s Session ID using XSS attack by using javascript.
- If an attacker sends a crafted link to the victim with the malicious JavaScript, when the victim clicks on the link, the JavaScript will run and complete the instructions made by the attacker.

Session Hijacking

- IP Spoofing
- Spoofing is pretending to be someone else.
- This is a technique used to gain unauthorized access to the computer with an IP address of a trusted host.
- In implementing this technique, attcker has to obtain the IP address of the client and inject his own packets spoofed with the IP address of client into the TCP session, so as to fool the server that it is communicating with the victim i.e. the original host.

Session Hijacking

- Blind Attack
- If attacker is not able to sniff packets and guess the correct sequence number expected by server, brute force combinations of sequence number can be tried.

Mitigation

- To defend a network with session hijacking, a defender has to implement both security measures at Application level and Network level.
- Network level hijacks can be prevented by Ciphering the packets so that the hijacker cannot decipher the packet headers, to obtain any information which will aid in spoofing.
- This encryption can be provided by using protocols such as IPSEC, SSL, SSH etc.
- Internet security protocol (IPSEC) has the ability to encrypt the packet on some shared key between the two parties involved in communication. IPsec runs in two modes: Transport and Tunnel.
- In Transport Mode only the data sent in the packet is encrypted while in Tunnel Mode both packet headers and data are encrypted, so it is more restrictive.

Iframe Injection

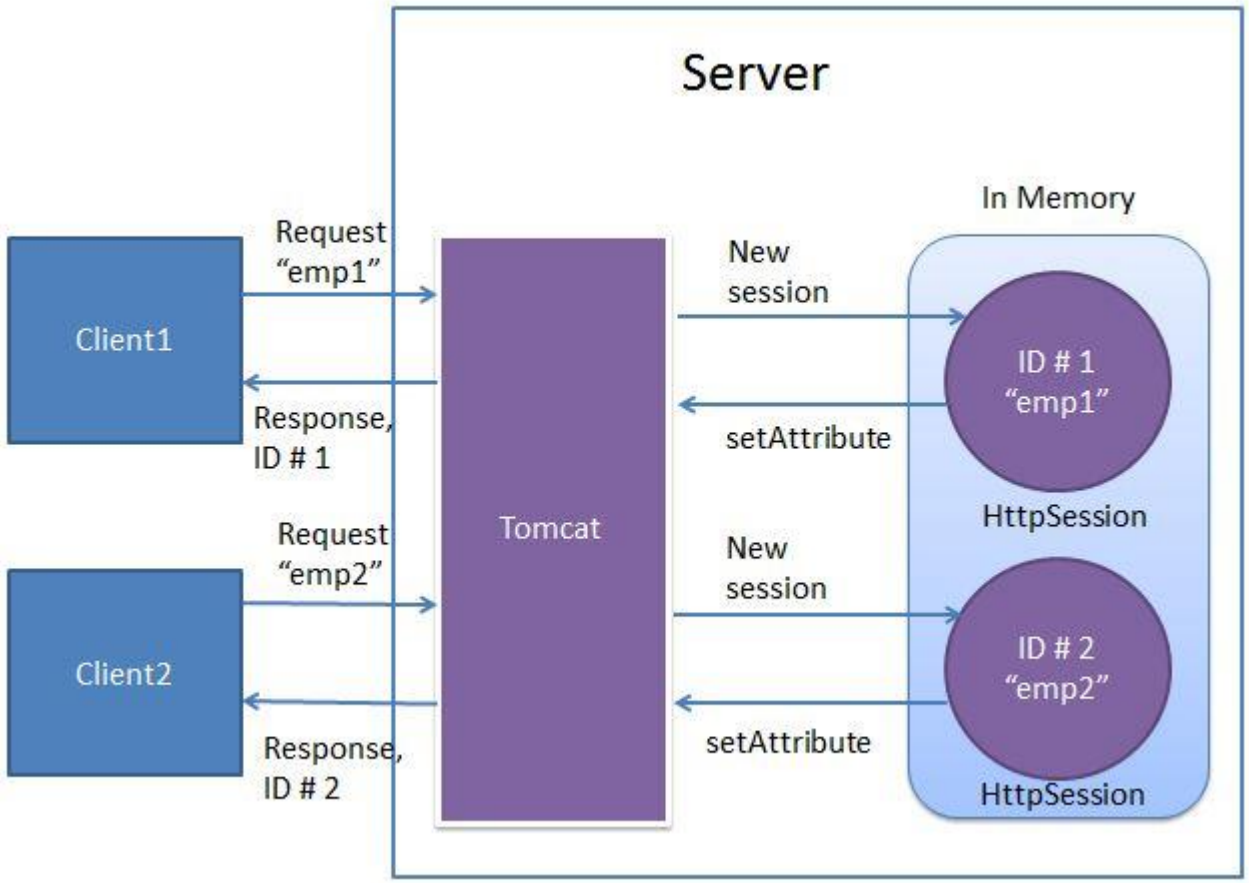
- A Frame Injection is a type of Code Injection vulnerability classified by OWASP Top 10 2017 in its A1 Injection category.
- Cross-site Scripting is naturally prioritized by bug bounty hunters since it seems easily exploitable and effective.
- But malicious hackers are also attracted to this vulnerability, because there are aspects of the Frame Injection attack that can allow hackers to redirect users to other malicious websites used for phishing and similar attacks.

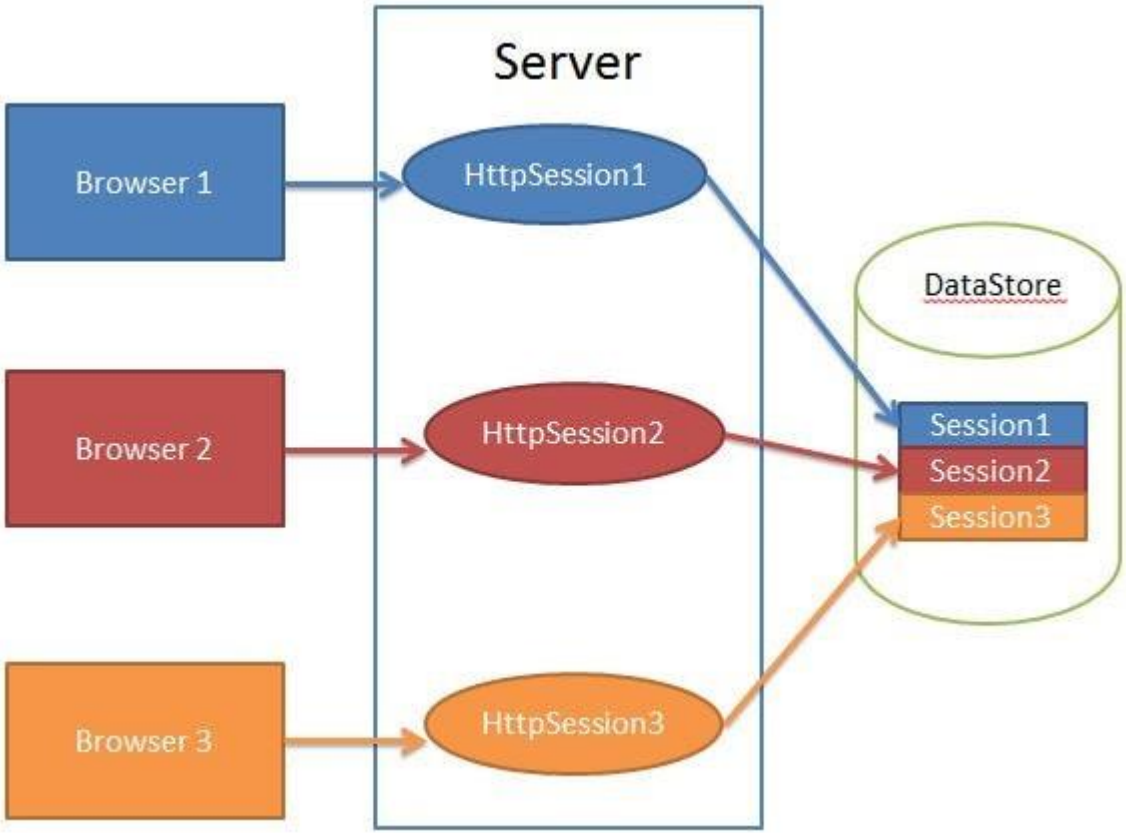
Iframe Injection

- 1. First of all attacker will find the Vulnerable websites using google dorks.
- 2. They test the vulnerability by inserting some iframe tag using the url.
- 3. then insert the Malicious Iframe code inside the webpage.
- For Example:
- he can insert this code using the url:
- <iframe src=”http://malwarewebpages/web.html” width=1 height=1 style=”visibility:hidden;position:absolute”></iframe>

IFrame Injection Mitigation

- Change your passwords of ftp, control panel and database.
- Inform to your hosting service about the injection attack and they will take care of server injection .
- Download all your files from the hosting and check whether they are infected or not. if you found any infected files, clean it.
- Buy a good antivirus software, Scan your Computer completely.
- Don’t use the Public systems for logging into your Hosting service.


Server
In Memory
Request
“emp1”
Client1
Response,
ID # 1
Request
“emp2”
Client2
Response,
ID # 2
Tomcat
New
session
setAttribute
New
session
setAttribute
ID # 1
“emp1”
HttpSession
ID # 2
“emp2”
HttpSession


Server
Browser 1
HttpSession1
DataStore
Browser 2
HttpSession2
Session1
Session2
Session3
Browser 3
HttpSession3

How to configure Maven Tomcat Plugin to use HTTPS (SSL/TLS)

Test

mvn tomcat7:run-war

Questions

Module Summary

- Spring Integration Framework.
- Message, Channel and Adapter
- Understood the different Component Integration
- Understood the Event-Driven Architecture