Quantitative Cyber-Security

Colorado State University

Yashwant K Malaiya

CS559

L27: Presentations

CSU Cybersecurity Center
Computer Science Dept

Presentations

- This is a research-oriented project. Please mention significant recent work and cite researchers and identify current trends challenges.
- Students with closely related presentations should coordinate among themselves to minimize overlap.
- Everyone: fill the peer-review form, and submit through canvas on
- Final: is two part
  - Final a: critial review of two specific project Final Reports
    - Assignment should be available Dec 10 and will be due on Dec 15.
  - Final b: proctored questions based (somewhat like midterm)
    - Dec 16 2-4 PM as scheduled. Perhaps 1 hour.

Presentations/Final Report

Th Dec 3, 2020

- Ravichandran, Shree Harini. Smartphone Security Model and Vulnerabilities
- Pineiro Rivera, Luis. Credit Card & Digital Wallet Security
- Padalia, Dhruv. Assessing effectiveness of Penetration Testing approaches
- Mulligan, Brett. Fuzzing Open Source IoT Project to Identify Novel Security Vulnerabilities
- Liu, Zijuan. Security in Virtualized Systems
- Kotian, Siddhi. Assessing Effectiveness of Penetration Testing approaches
- Zhao, Qingyi. Quantitative examination of phishing (moved)

RISK ASSESSMENT: CREDIT CARD AND DIGITAL WALLET SECURITY

Luis E Pineiro Rivera

CS559 – Quantitative security

Dec 3 2020

OVERVIEW

- Research Goals
- Online Credit Card Payment Protocols
- Risk Assessment Model
- Credit Cart Payment Frameworks Around the Globe

RESEARCH GOALS

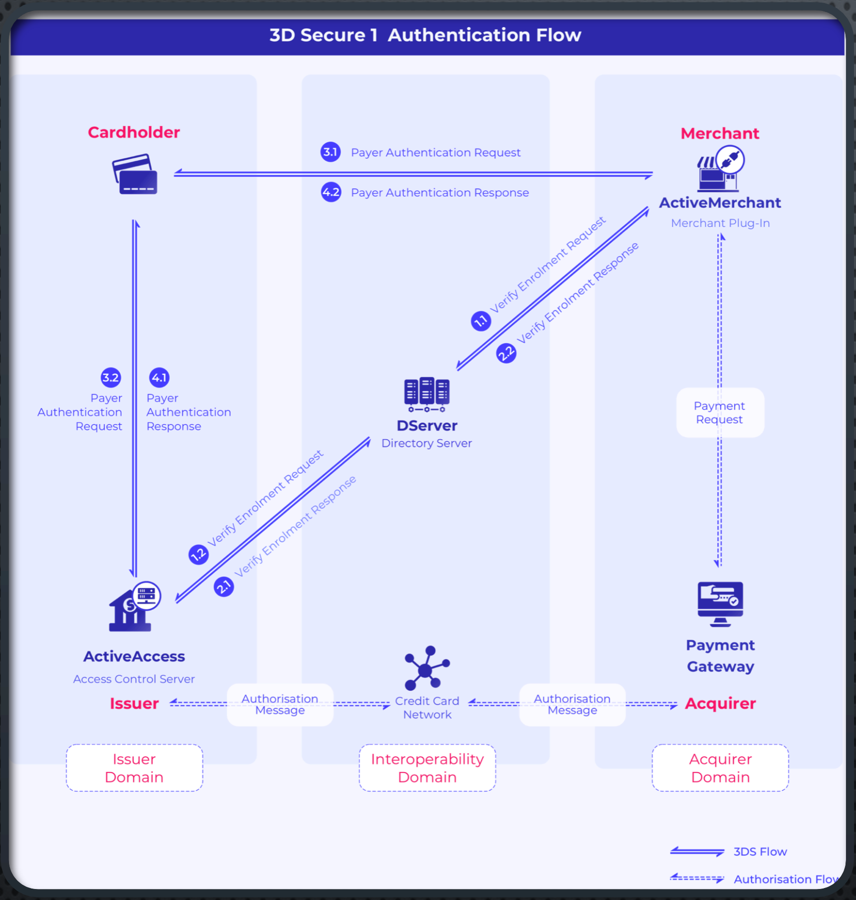
- Get out of my comfort zone
- Identify current Technologies
- Risk Assessment Analysis Process
- Lear about the standards across the globe

ONLINE CREDIT CARD PAYMENT PROTOCOLS

- 3D Secure version 1
- 3D Secure version 1
- Digital Wallet

3D Secure Version 1

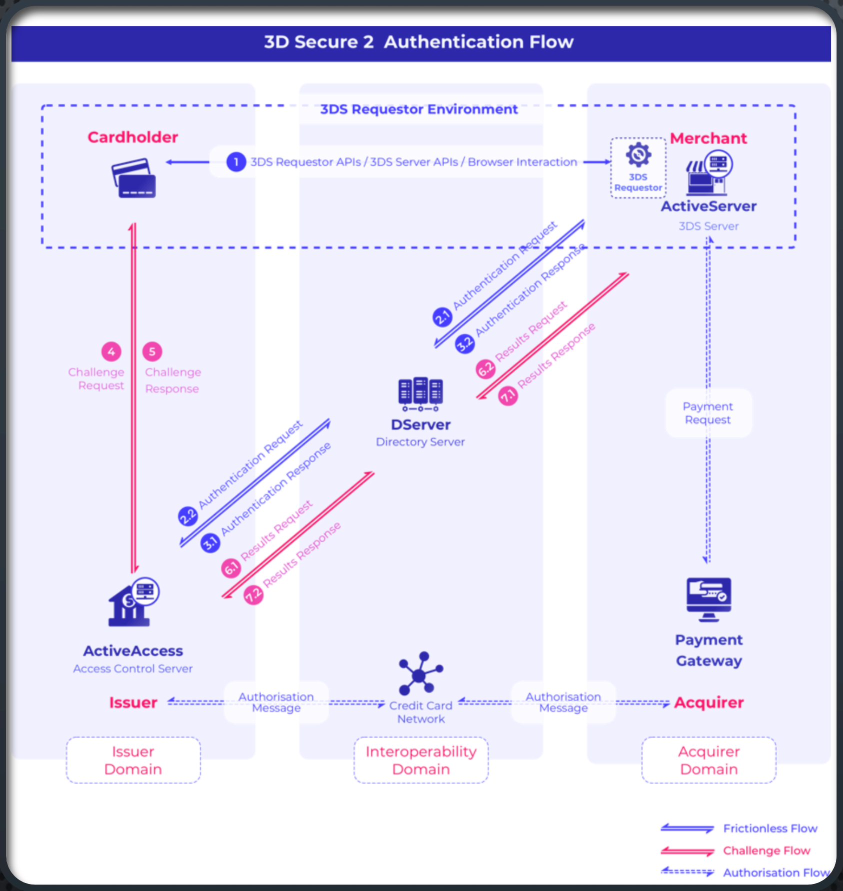
- Established in 2000
- Provides credit card authentication through credit card account login window
  - Visa: Verified by Visa
  - Amex: SafeKey
  - Discover: MasterCard Secure Code
- Uses proprietary authentication protocol and server to validate transaction
- Cons:
  - Login Pop-up Window
  - Can be used by malicious actor to grab Credit Cart username and password



3D SECURE VERSION 1

3D Secure Version 2

- Established in 2015
- Frictionless Flow: No more authentication window
- Reduces Cart Abandonment
- Additional standards to comply with European requirements
- New Features
  - Additional info collected during each transaction by merchant
    - IP, MAC address, PC HW Info etc.
  - To be used by bank to authenticate validity of purchase (risk Model)
  - If bank deems purchase questionable, the user will be challenged



3D SECURE VERSION 2

DIGITAL WALLET – APPLE PAY

- iOS device becomes the card
- Credit Card information is stored in Secure Element (SE) chip of the device
  - Only user has access to this information and not Apple
  - SE Chip – Common Standard

DIGITAL WALLET – APPLE PAY

- Is it Secure?
  - User enrolls credit card in Digital Wallet
  - Issuing Bank approves
  - Bank will create unique Device Account Number
  - Encrypted information will be stored in SE chip
  - No Credit Card information is stored on the actual device
  - Only Bank can decrypt this information

DIGITAL WALLET – APPLE PAY

- How does it work?
  - It uses NFC or Apple Pay API
  - iOS device will request user authentication (Face ID, Touch ID or passcode)
  - SE Chip generates a token and send it along with unique Device Account Number
  - Bank decrypts token and verifies device account number to see if they match.
- What about Online Payments?
  - Via Apple Pay API
  - Apple will encrypt Token and Device Account Number using developer/bank key
  - Only the developer or bank can decrypt this information
  - Token and Device Account Number will be sent to Bank for decryption and authorization

Risk Assessment Model

- How do we assess the risk related to each payment model?
  - Create risk Types and assess weighted impact
    - Merchant Risk – 30%
    - User Risk – 30%
    - Transaction Risk – 20%
    - Vulnerability Risk – 20%
  - Generate Risk Values (scale 1 to 10)
    - Very Low – 1
    - Low – 3
    - Medium – 5
    - High – 8
    - Very High 10
- Risk Assessment Formula:
  - Risk = (MR*I) + (UR*I) + (TR*I) + (VR*I)

RISK ASSESSMENT MODEL

- Risk Assessment

System	MR	UR	TR	VR
3DS1	Medium	High	Medium	High
3DS2	Very Low	Medium	Low	Low
Apple Pay	Very Low	Very Low	Very Low	Very Low

- Apply Risk Formula

System	MR	UR	TR	VR	Risk
3DS1	1.5	2.4	1	1.6	6.5
3DS2	.3	1.5	.2	.2	2.2
Apple Pay	.3	.3	.2	.2	1

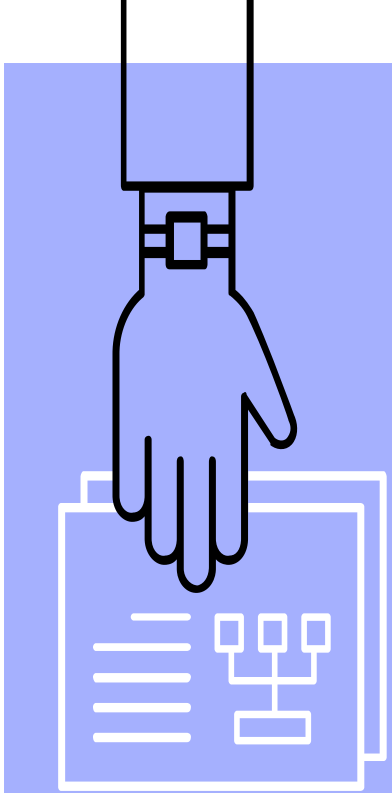
- Results
  - 3DS1 – Medium to High Risk
  - 3DS2 – Very Low to Low Risk
  - Apple Pay – Low Risk

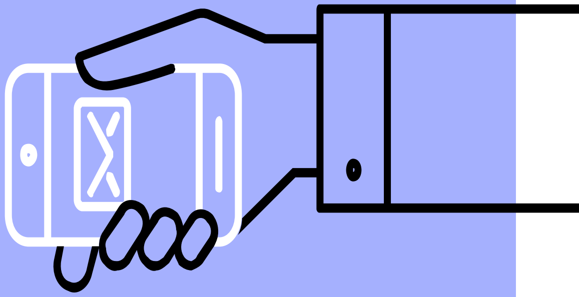
FRAMEWORKS AROUND THE GLOBE

- India - PaySecure
- China – UnionPay Online Payments (UPOP)
- Russia - MIR
- Europe – Directive on Payment Services (PSD2)

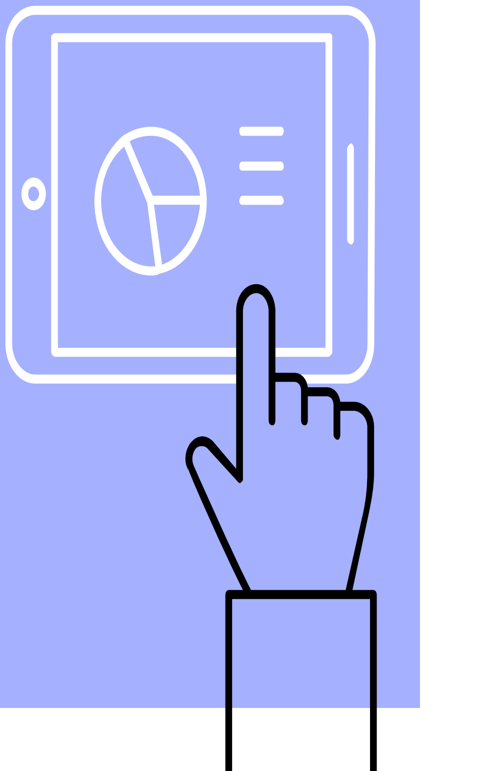
SUMMARY

- Research Goals
- Online Credit Card Payment Protocols
- Risk Assessment Model
- Credit Cart Payment Frameworks Around the Globe



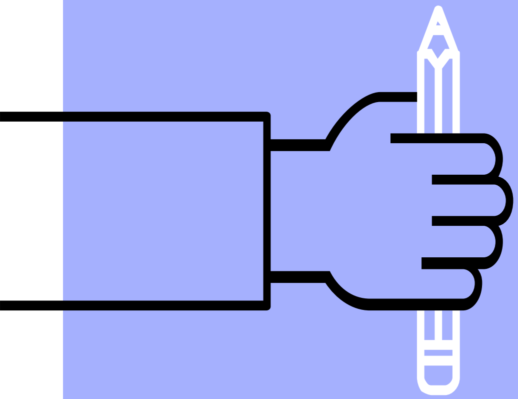


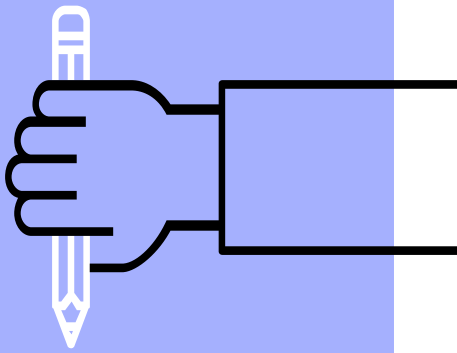
Assessing effectiveness of Penetration Testing Approaches



By - Dhruv Padalia

CS559 - Colorado State University

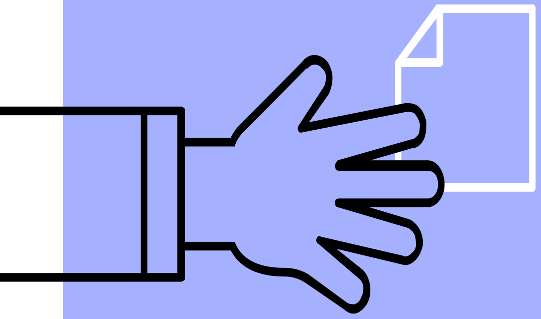




1.
Introduction

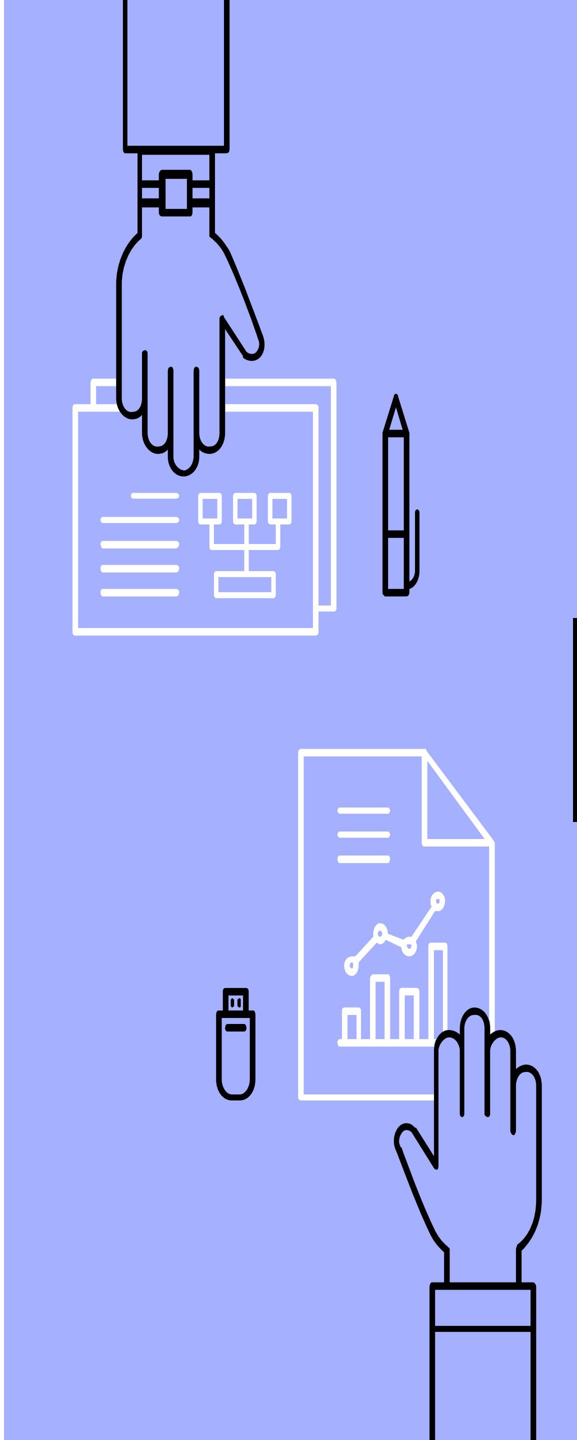
What is penetration testing?

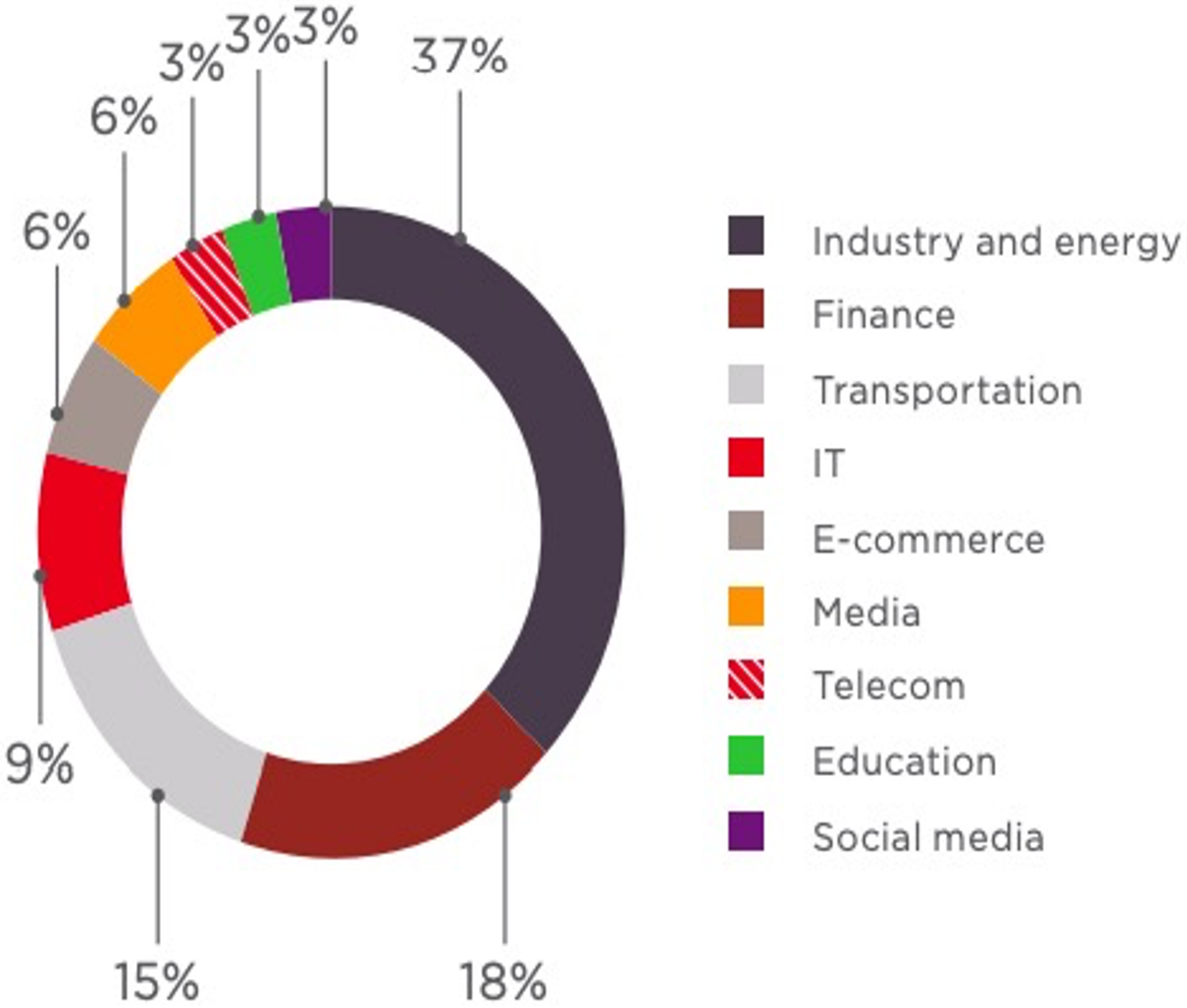
Internal vs External penetration testing



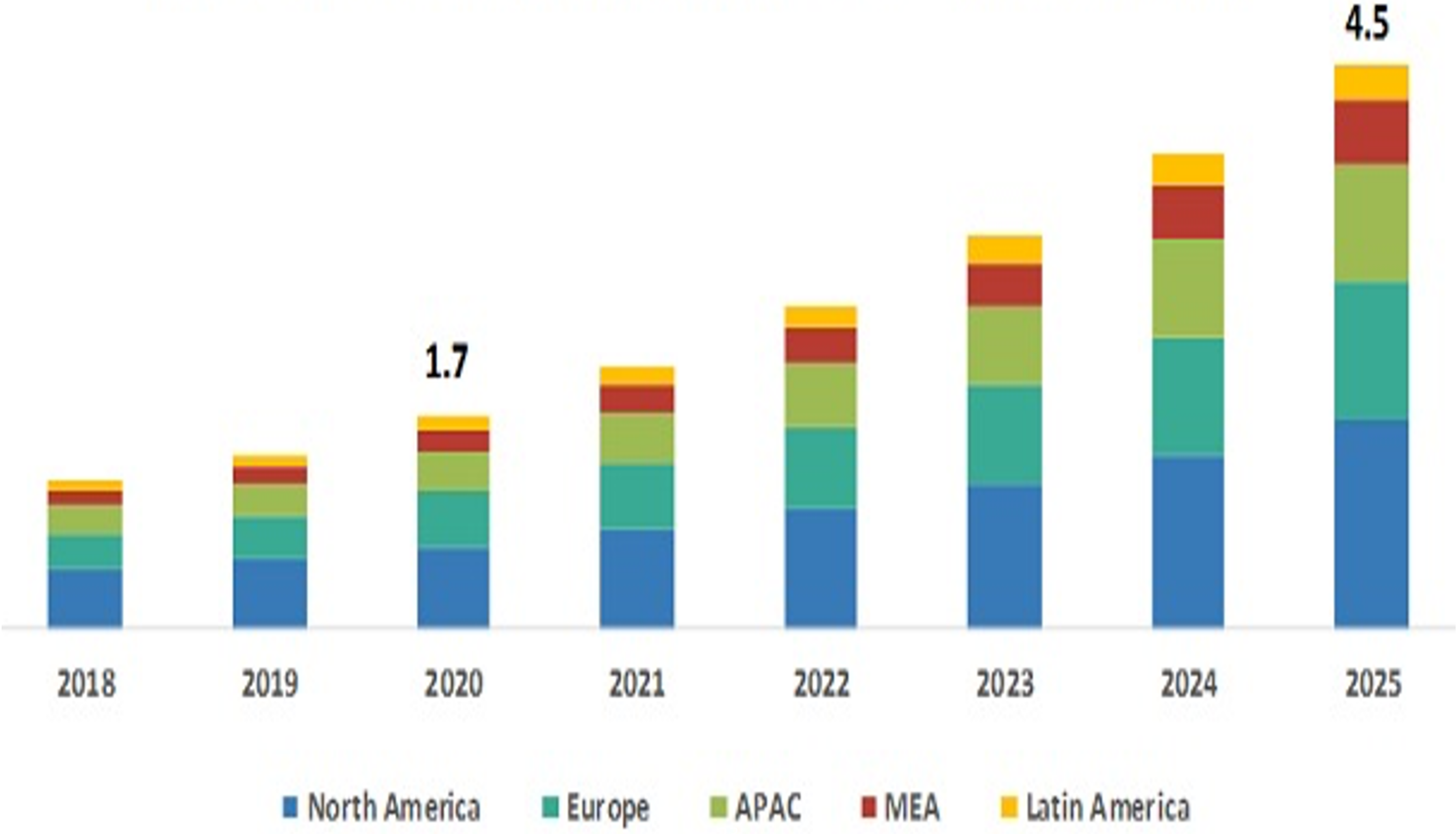
What is Penetration Testing?

- Simulated cyber attack
- Types of penetration testing
  - Network
  - Web application
  - Client side
  - Wireless
  - Social Engineering
  - Physical Access




3%
3%
3%
37%
6%
6%
9%
15%
18%
Industry and energy
Finance
Transportation
IT
E-commerce
Media
Telecom
Education
Social media

[redacted]



PENETRATION TESTING MARKET, BY REGION (USD BILLION)

Source: Mark[redacted]kets Analysis

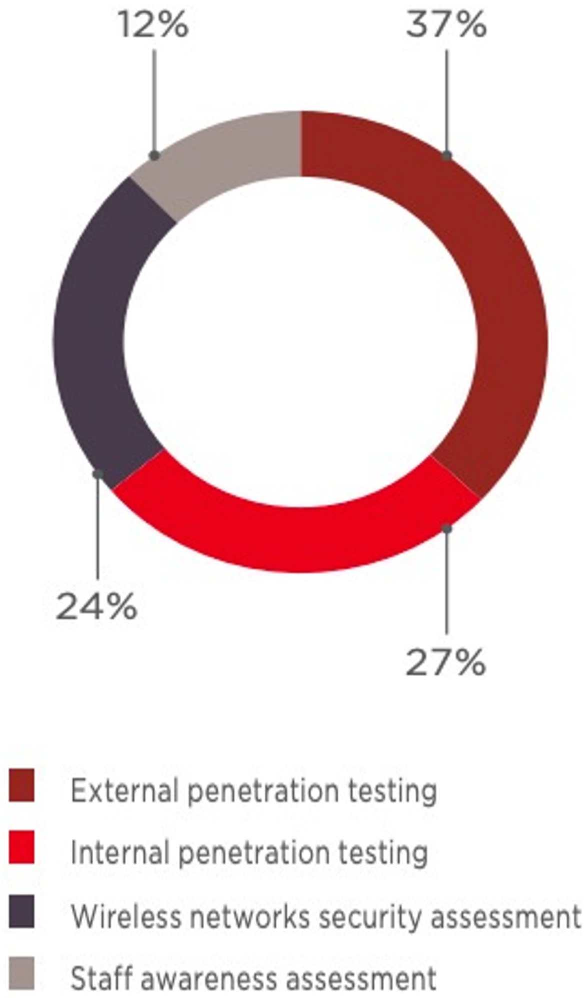
External Vs Internal Penetration Testing

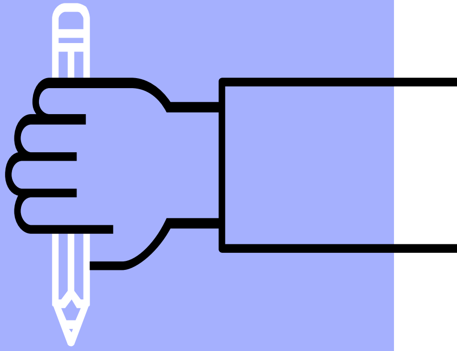
External

- Targets assets visible on the internet
- Example - company website, email, DNS
- Gain access and extract valuable data

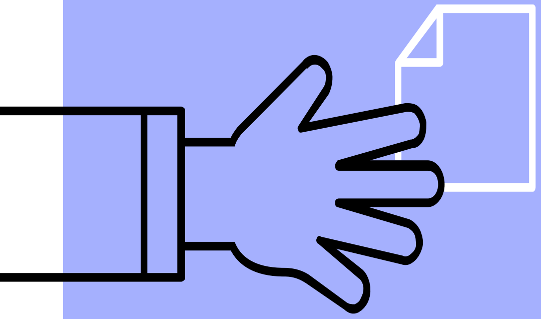
Internal

- A tester with access to an application behind its firewall



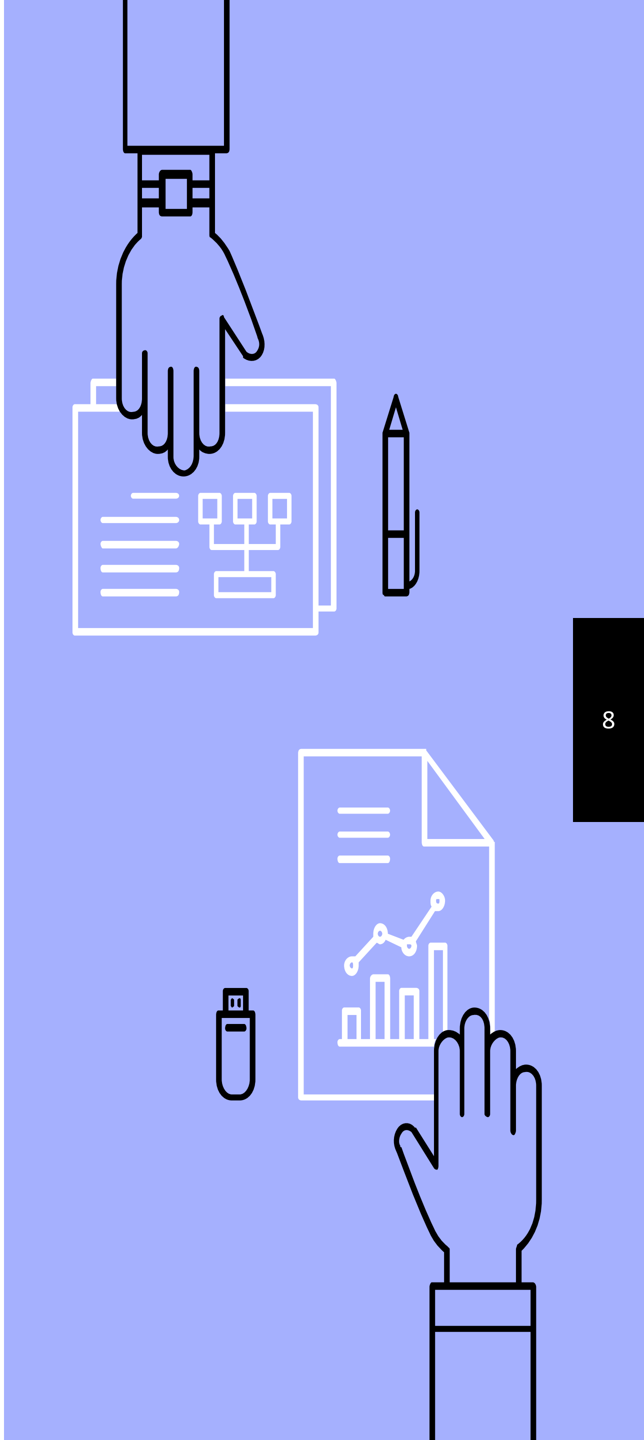


2.
External Penetration Testing



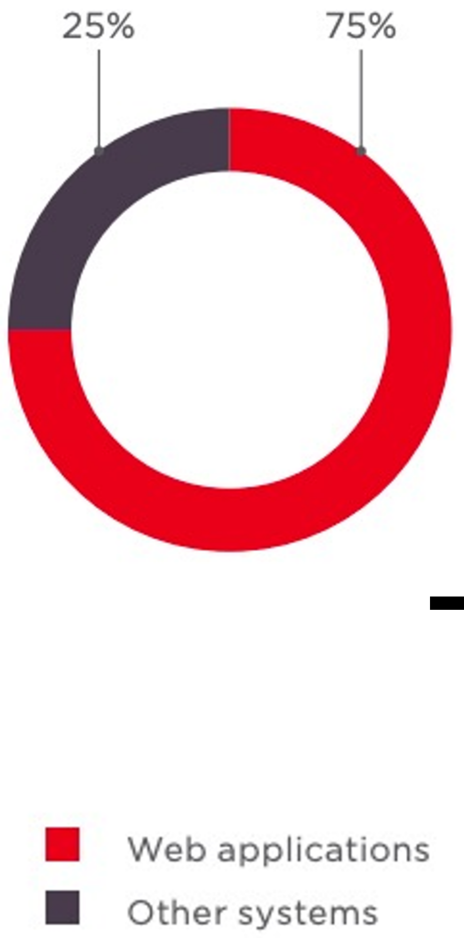
External Penetration Testing: Insights

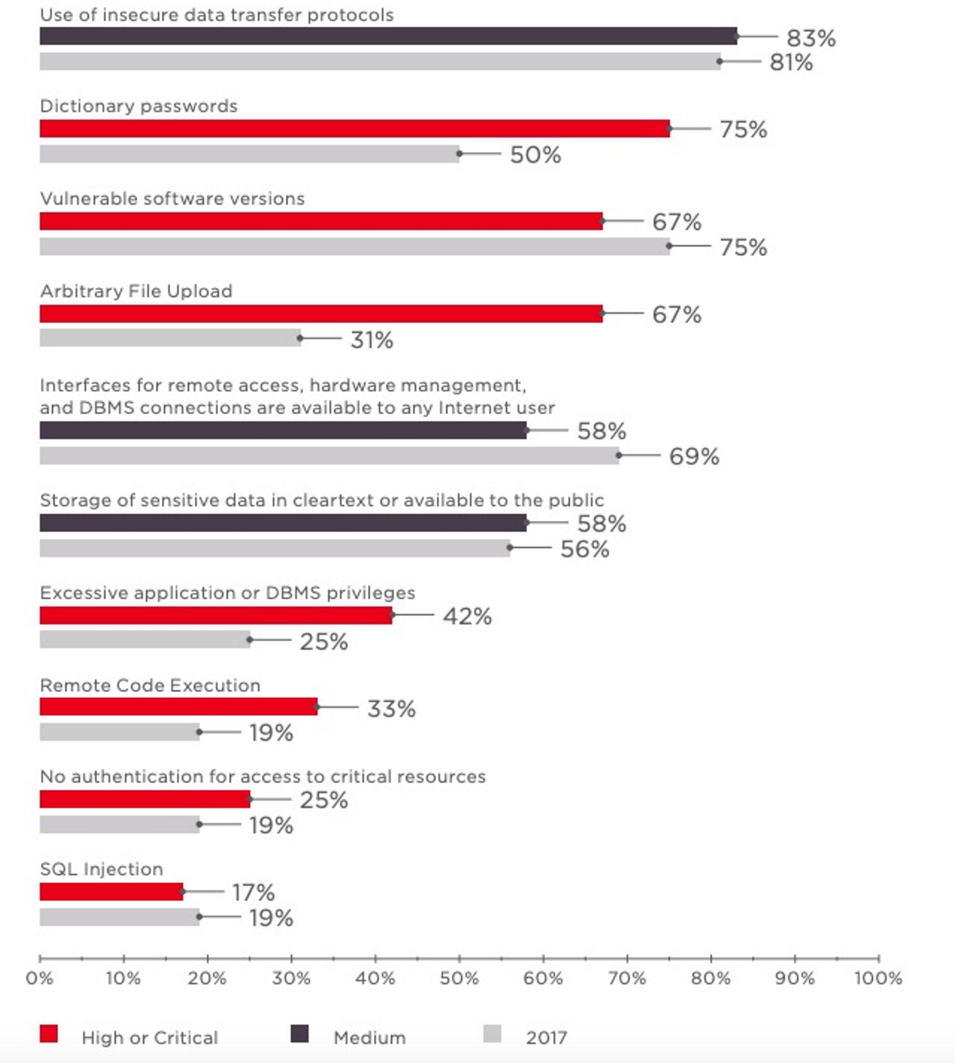
- Attempt to gain access and get valuable data
- In 2018, 92% of the companies was breached during external pentesting



External Penetration Testing: Causes

- Attempt to gain access and get valuable data
- In 2018, 92% of the companies was breached during external pentesting
- 75% due to poor web application

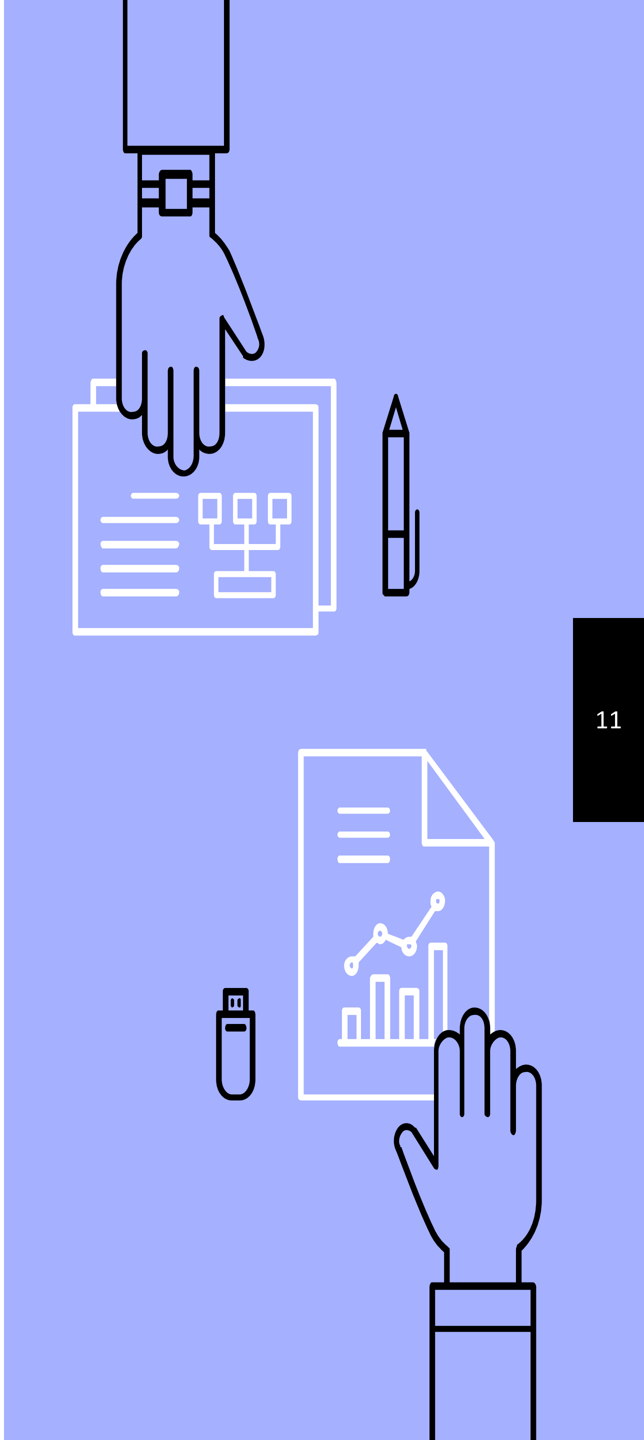



Use of insecure data transfer protocols
83%
81%
Dictionary passwords
75%
50%
Vulnerable software versions
67%
75%
Arbitrary File Upload
67%
31%
Interfaces for remote access, hardware management, and DBMS connections are available to any Internet user
58%
69%
Storage of sensitive data in cleartext or available to the public
58%
56%
Excessive application or DBMS privileges
42%
25%
Remote Code Execution
33%
19%
No authentication for access to critical resources
25%
19%
SQL Injection
17%
19%
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
High or Critical
Medium
2017

[redacted]

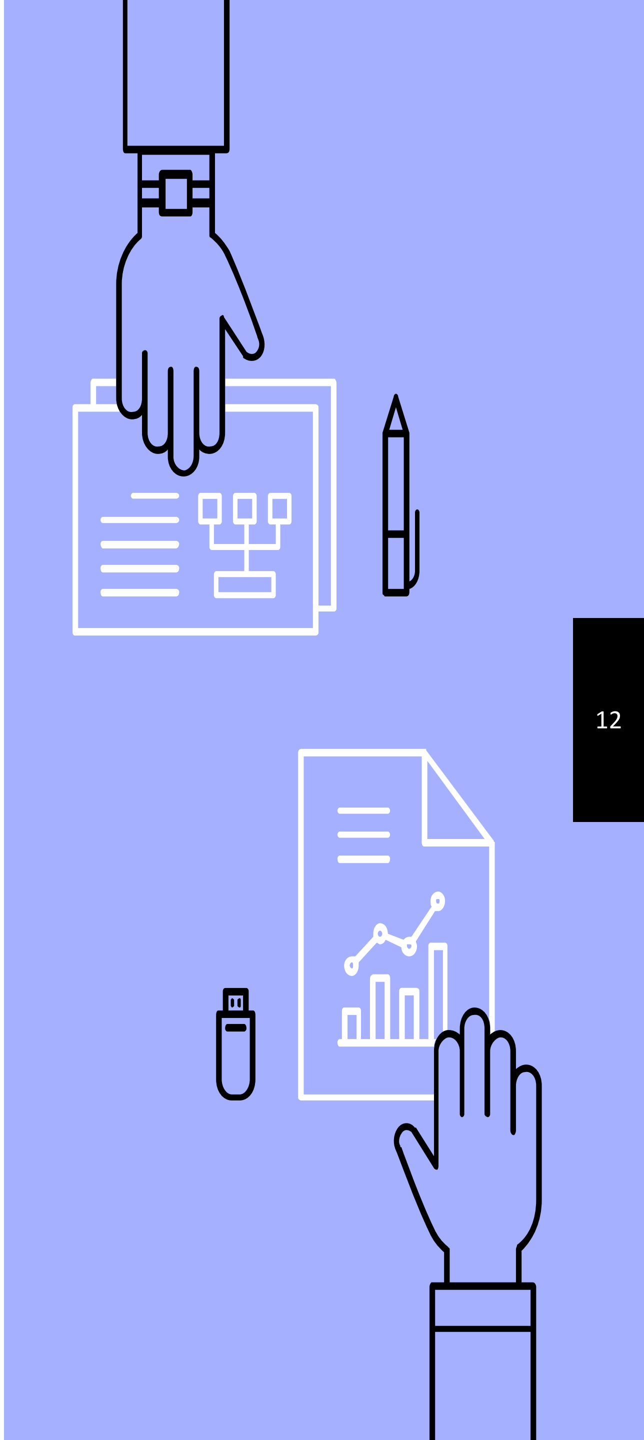
External Penetration Testing: Tools used

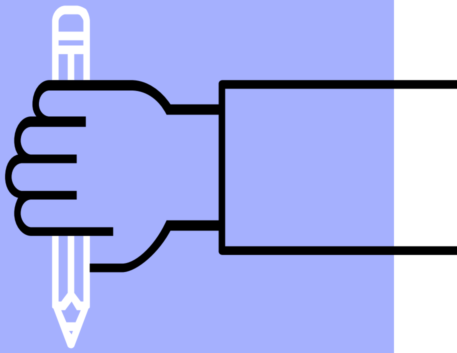
- Injection: Manually, Sqlmap, DSSS
- Password Cracking: Hashcat, John the ripper
- Protocol testing: tcpdump, wireshark



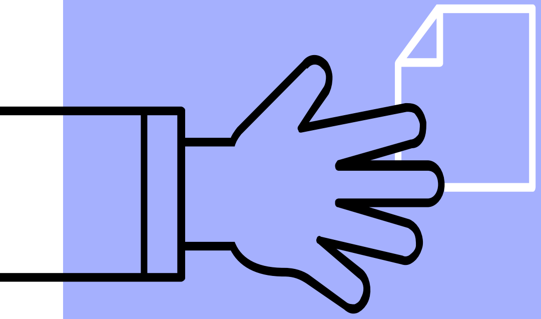
External Penetration Testing: Remedies

- Enforce strict password policies
- Web application testing using tools like OWASP ZAP
- Use secure data transfer protocol



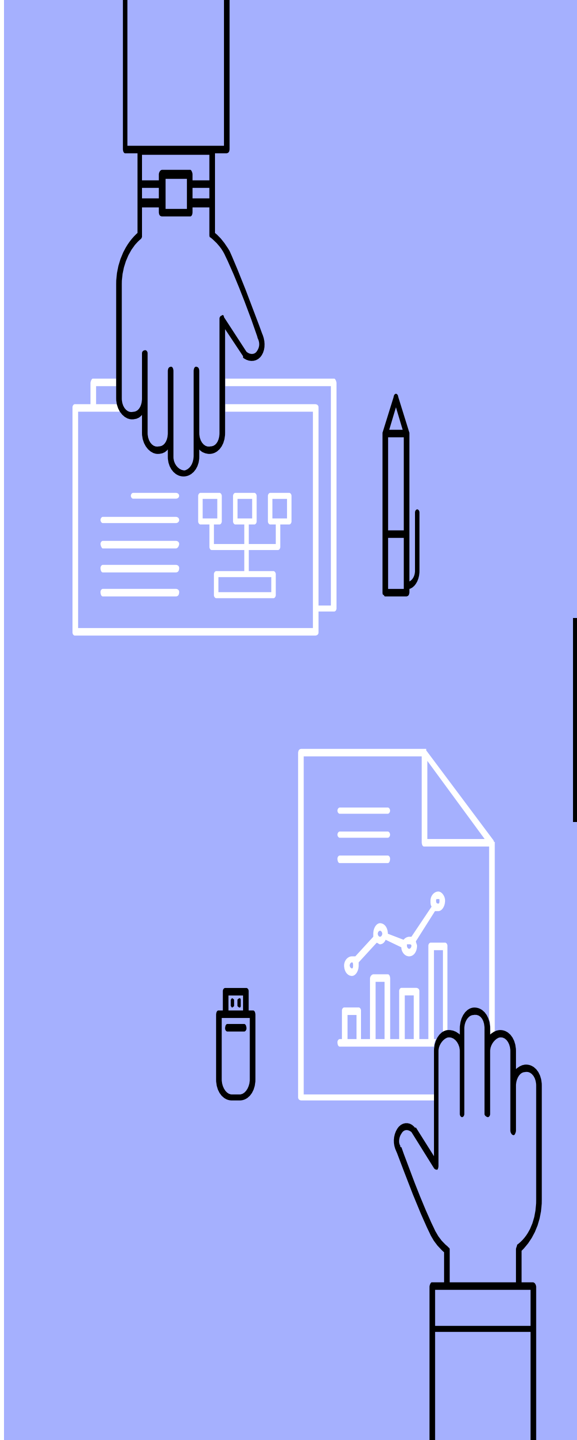


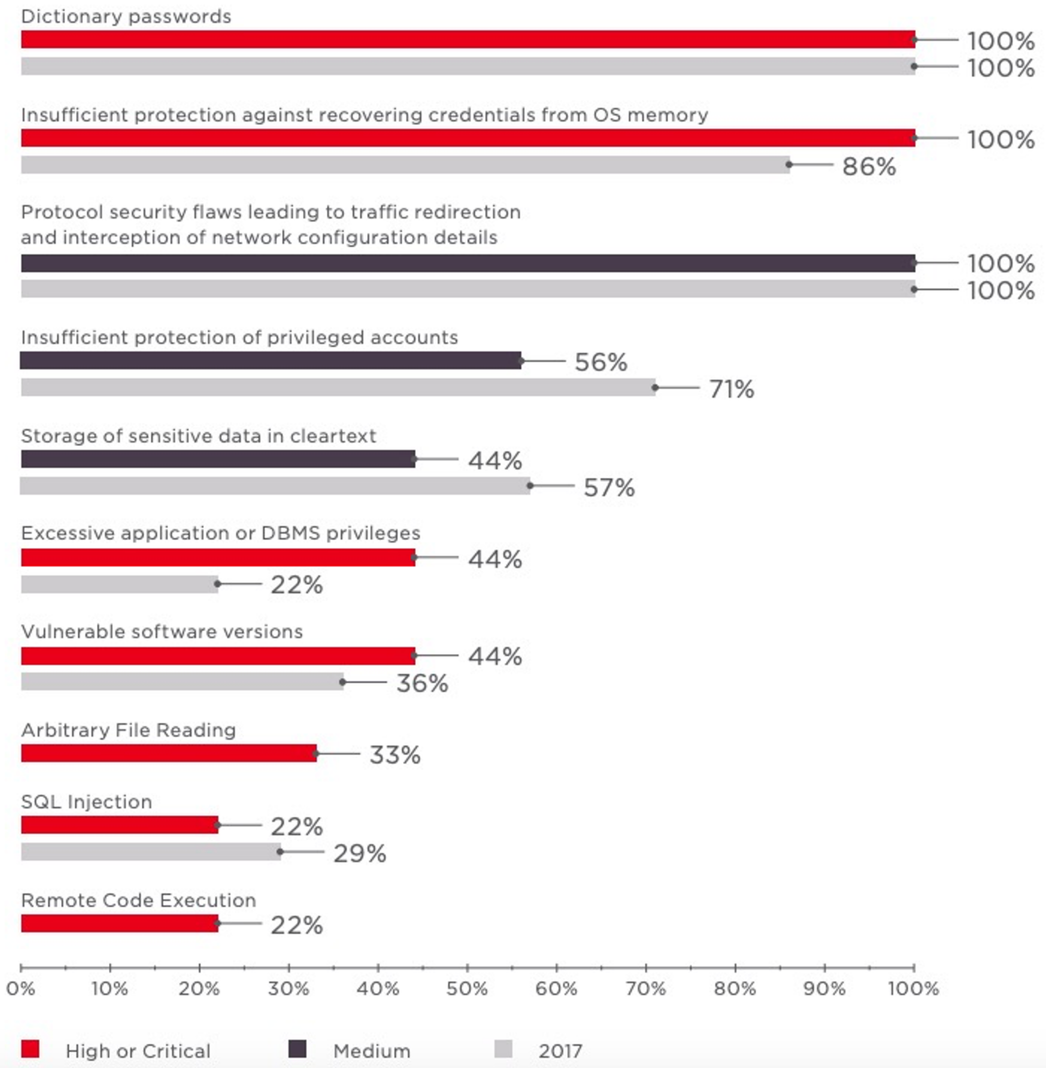
4.
Internal Penetration Testing



Internal Penetration Testing: Insights

- Gaining full control of infrastructure
- In 2018, 100% of the companies was breached during internal pentesting

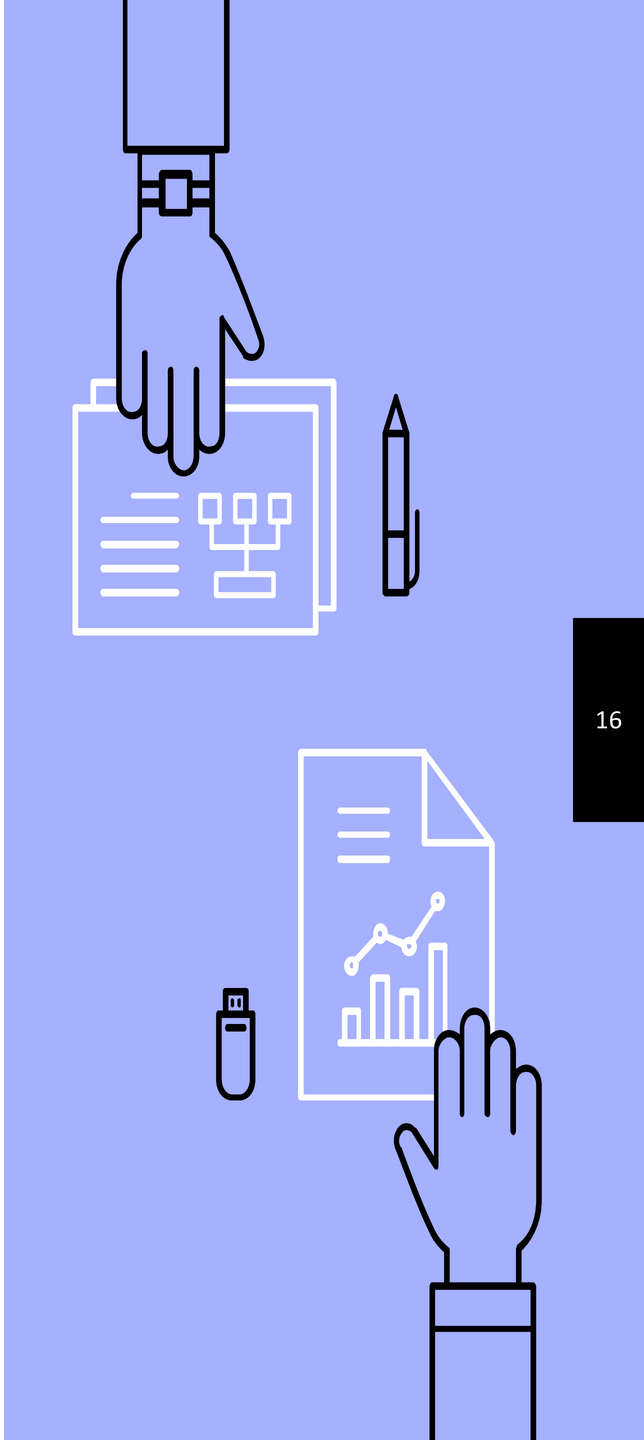



Dictionary passwords
100%
100%
Insufficient protection against recovering credentials from OS memory
100%
86%
Protocol security flaws leading to traffic redirection and interception of network configuration details
100%
100%
Insufficient protection of privileged accounts
56%
71%
Storage of sensitive data in cleartext
44%
57%
Excessive application or DBMS privileges
44%
22%
Vulnerable software versions
44%
36%
Arbitrary File Reading
33%
SQL Injection
22%
29%
Remote Code Execution
22%
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
High or Critical
Medium
2017

[redacted]

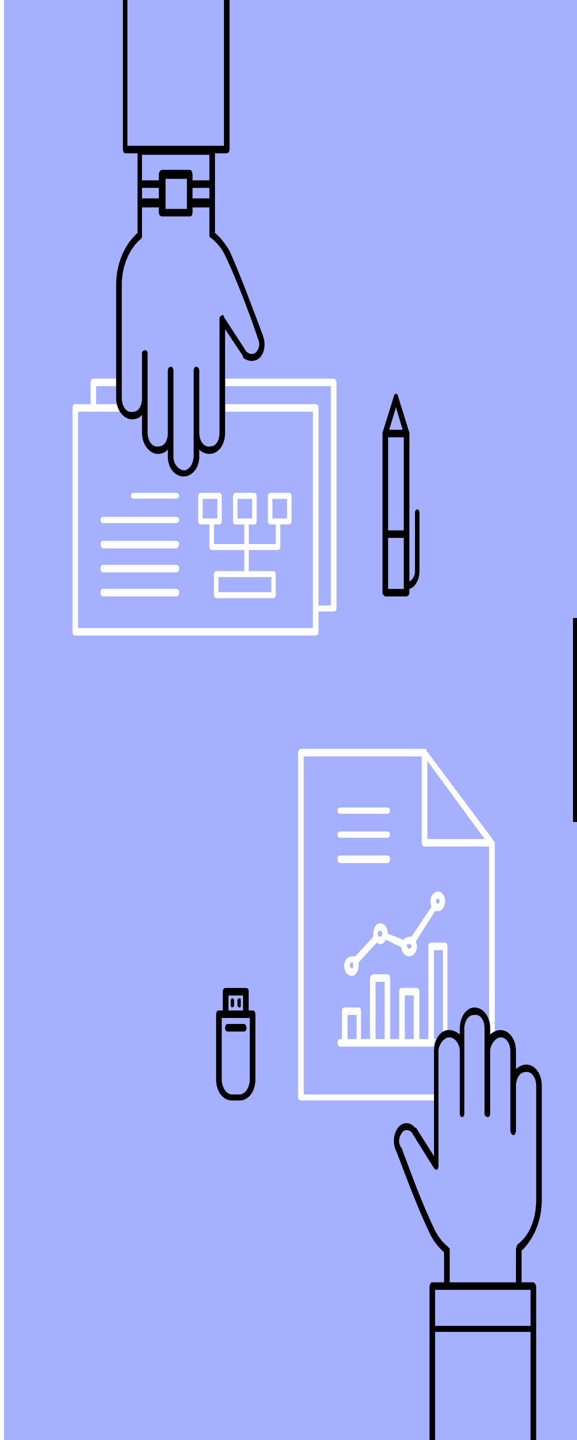
Internal Penetration Testing: Tools used

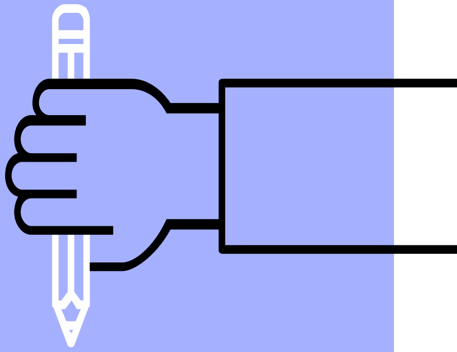
- Injection: Manually, Sqlmap, DSSS
- Password Cracking: Hashcat, John the ripper
- Open Ports: nmap, masscan



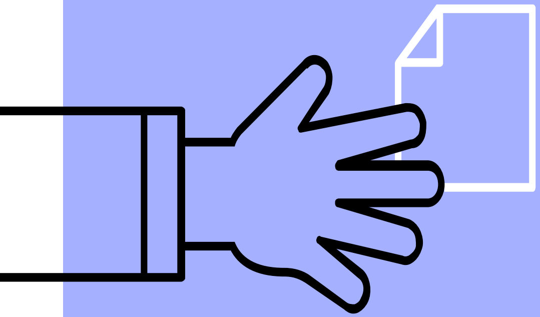
Internal Penetration Testing: Remedies

- Enforce strict password policies
- Close unused ports





5.
Tools



Tools

Sqlmap

Tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers

Nmap

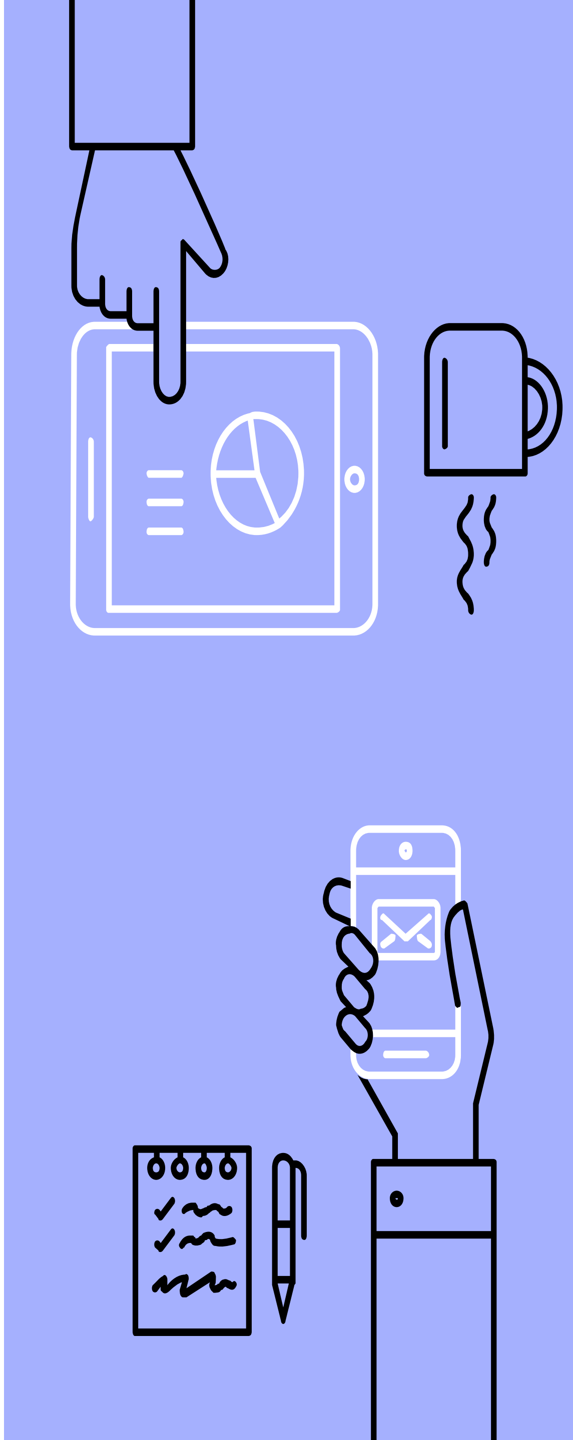
Network Mapper is a network discovery and security auditing tool

DSSS

Damn Small SQLi Scanner is a SQL injection vulnerability scanner written in under 100 lines of code.

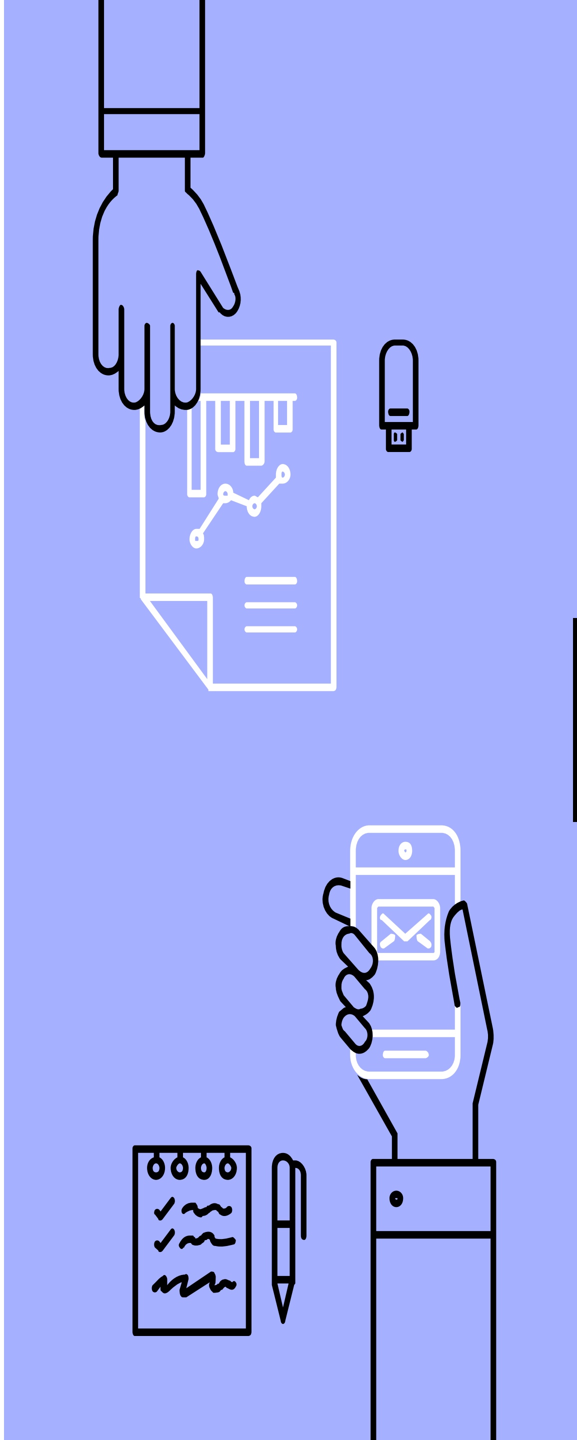
Masscan

Masscan is a internet port scanner



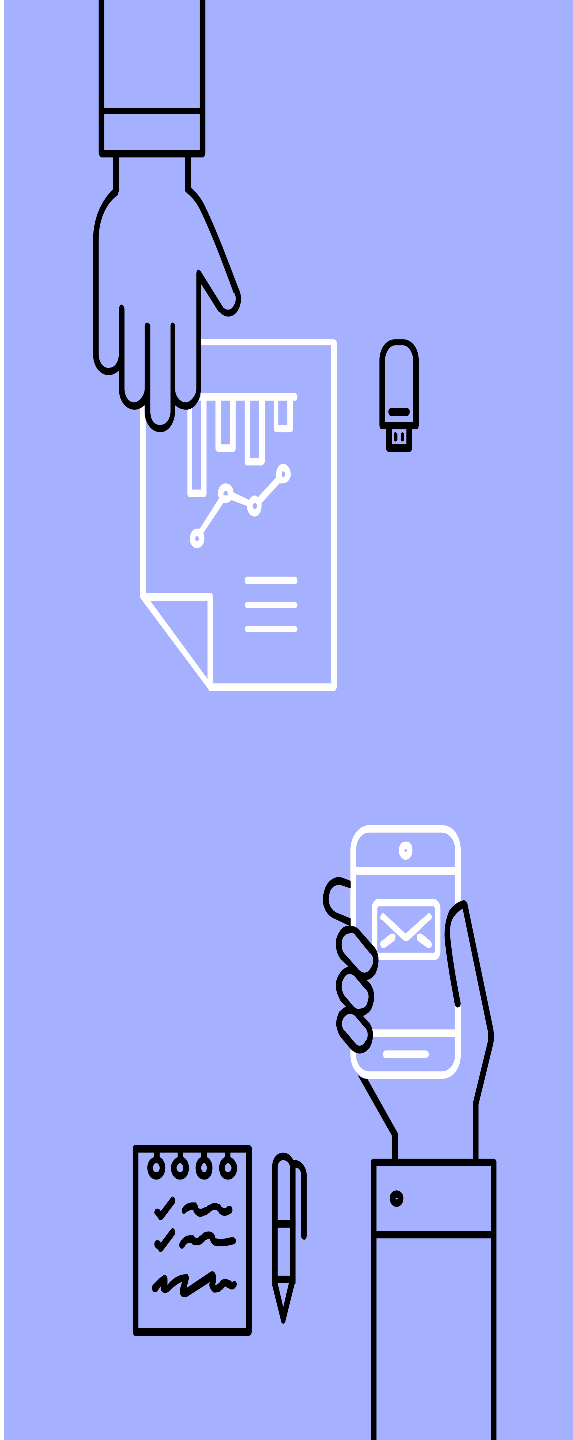
Nmap vs masscan

	Time Taken	CPU Utilization	Scans TCP and UDP protocols
Nmap	11.3	0%	yes
Masscan	4.06	2%	yes



Sqlmap vs DSSS

	Time Taken	CPU Utilization	Successful Detection
SqlMap	19 s	11%	yes
DSSS	2.9 s	6%	no



THANKS!

Any questions?

References

[1] Positive Techology, https://www.ptsecurity.com/ww-en/analytics/corp-vulnerabilities-2019/

[2] Kumar R., Tlhagadikgora K. (2019) Internal Network Penetration Testing Using Free/Open Source Tools: Network and System Administration Approach. In: Luhach A., Singh D., Hsiung PA., Hawari K., Lingras P., Singh P. (eds) Advanced Informatics for Computing Research. ICAICR 2018. Communications in Computer and Information Science, vol 956. Springer, Singapore. https://doi.org/10.1007/978-981-13-3143-5_22

[3] S.P. Ganesh and G. Anandhi, “Database Security: A Study on Threats And Attacks”, International Journal on Recent and Innovation Trends in Computing and Communication, vol. 4(6), pp. 512-513, 2015.

[4] P. Shi, F. Qin, R. Cheng and K. Zhu, "The Penetration Testing Framework for Large-Scale Network Based on Network Fingerprint," 2019 International Conference on Communications, Information System and Computer Engineering (CISCE), Haikou, China, 2019, pp. 378-381, doi: 10.1109/CISCE.2019.00089.

[5] H. M. Z. A. Shebli and B. D. Beheshti, "A study on penetration testing process and tools," 2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT), Farmingdale, NY, 2018, pp. 1-7, doi: 10.1109/LISAT.2018.8378035.

[6] PurpleSec, https://purplesec.us/firewall-penetration-testing/, website.

[7] Campuquip, https://www.compuquip.com/blog/5-firewall-threatsand-vulnerabilities-to-look-out-for, website.

[8] nmap: network port scanner, https://nmap.org/

[9] DSSS, damn small sqli scanner, https://github.com/stamparm/DSSS

[10] sqlmap: SQL Injectejction, http://sqlmap.org/

[11] Five Types of Penetration Test to Zero in Potential Vulnerabilities, https://www.techbeamers.com/penetration-test-and-types/

[redacted]

Fuzzing Open Source IoT Projects

Brett Mulligan

CS559

Colorado State University

Overview

- Motivation
- Methods
- Results
- Lessons Learned
- Conclusion

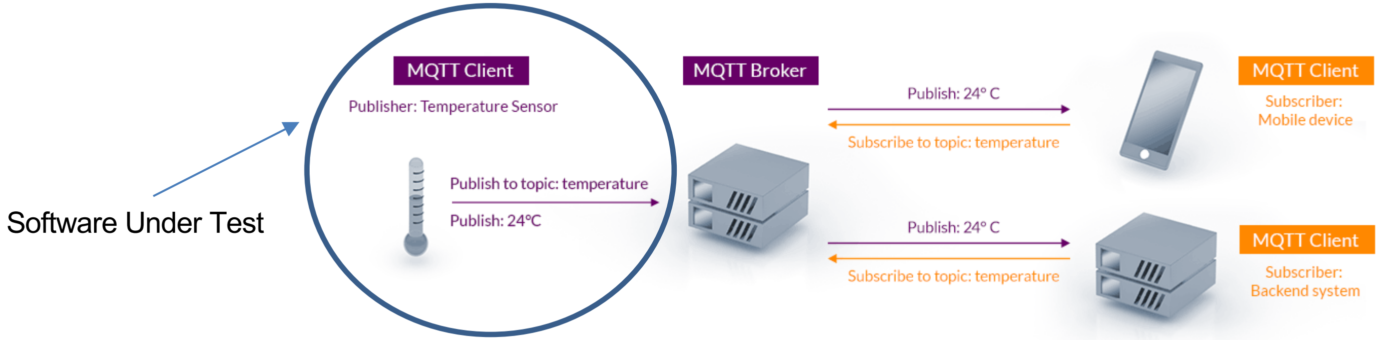
Motivation: IoT Still on the Rise

- Estimated 75 billion IoT connected devices by 2025 [6]
- Phones, mesh networks, sensor networks
- Home automation
- Swarms & fleets
- Popular botnet target [7]

https://www.counterpointresearch.com/iot-world-2018-key-oems-trends-analysis/

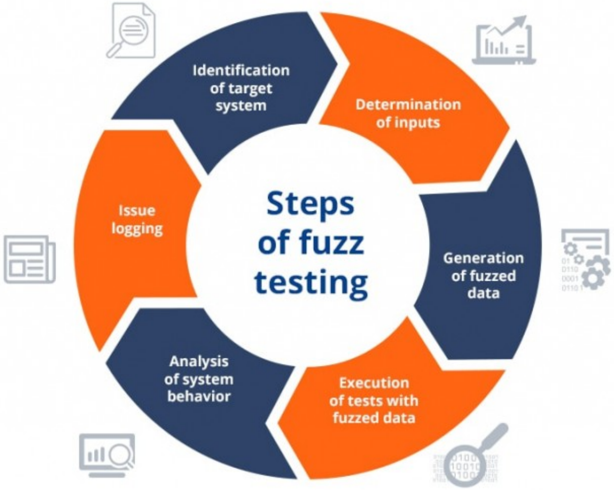
Motivation: MQTT

- Message Queuing Telemetry Transport
- Common IoT Protocol [4]
- Designed for low bandwidth, low power, and unreliable connectivity
- Subscriber/Publisher model



Motivation: Fuzzing

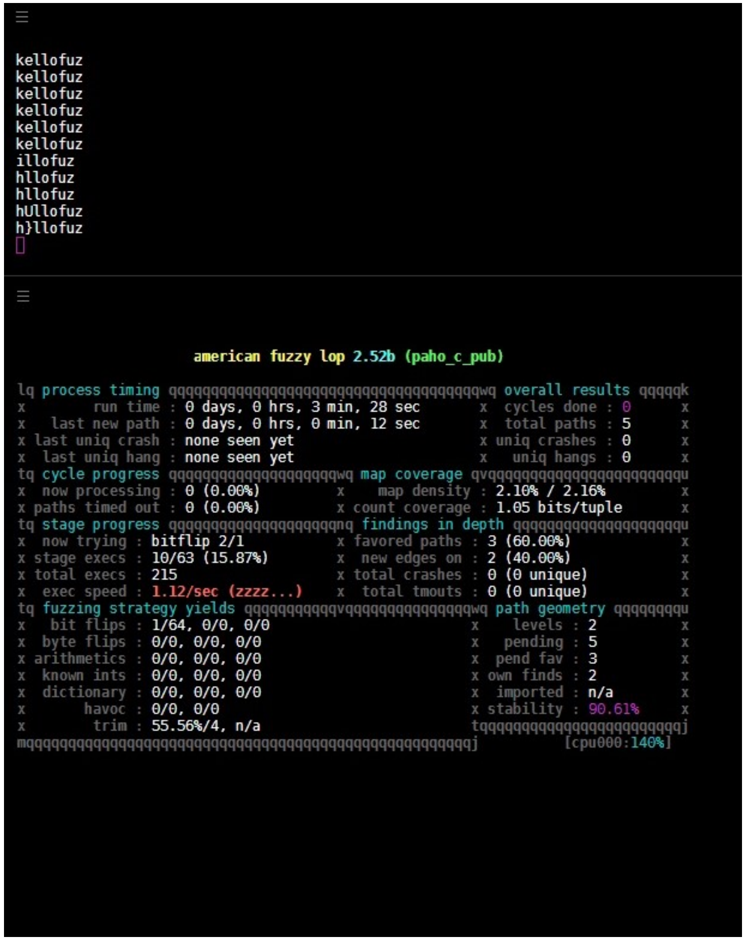
- Fuzzing is a testing technique for finding vulnerabilities in software applications by sending unexpected input data to target systems and then monitoring the results. [2]
- American Fuzzy Lop's proven record of finding real vulnerabilities: OpenSSL, Safari, etc.
- Target selected:
  - Eclipse Foundation's Paho MQTT Library


Steps of fuzz testing
Identification of target system
Determination of inputs
Generation of fuzzed data
Execution of tests with fuzzed data
Analysis of system behavior
Issue logging

https://blog.qatestlab.com/2011/03/10/what-is-fuzz-testing/

Methods

- AFL-Fuzz – grey box, black box (dumb)
- Radamsa – black box input generation
- Varied input generation
    - HTML
    - Text file
    - PNG
    - PDF
    - Radamsa mutation (text)
- Mosquitto broker, monitor outputs



```
kellofuz
kellofuz
kellofuz
kellofuz
kellofuz
kellofuz
illofuz
hllofuz
hllofuz
hUllofuz
h}llofuz
```

```
                     american fuzzy lop 2.52b (paho_c_pub)

lq process timing qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqwq overall results qqqqqk
x        run time : 0 days, 0 hrs, 3 min, 28 sec       x  cycles done : 0      x
x   last new path : 0 days, 0 hrs, 0 min, 12 sec       x  total paths : 5      x
x last uniq crash : none seen yet                      x uniq crashes : 0      x
x  last uniq hang : none seen yet                      x   uniq hangs : 0      x
tq cycle progress qqqqqqqqqqqqqqqqqqqqwq map coverage qvqqqqqqqqqqqqqqqqqqqqqqqu
x  now processing : 0 (0.00%)        x    map density : 2.10% / 2.16%          x
x paths timed out : 0 (0.00%)        x count coverage : 1.05 bits/tuple        x
tq stage progress qqqqqqqqqqqqqqqqqqqqnq findings in depth qqqqqqqqqqqqqqqqqqqqu
x  now trying : bitflip 2/1          x favored paths : 3 (60.00%)              x
x stage execs : 10/63 (15.87%)       x  new edges on : 2 (40.00%)              x
x total execs : 215                  x total crashes : 0 (0 unique)            x
x  exec speed : 1.12/sec (zzzz...)   x  total tmouts : 0 (0 unique)            x
tq fuzzing strategy yields qqqqqqqqqqqvqqqqqqqqqqqqqqqwq path geometry qqqqqqqqu
x   bit flips : 1/64, 0/0, 0/0                         x    levels : 2           x
x  byte flips : 0/0, 0/0, 0/0                          x   pending : 5           x
x arithmetics : 0/0, 0/0, 0/0                          x  pend fav : 3           x
x  known ints : 0/0, 0/0, 0/0                          x own finds : 2           x
x  dictionary : 0/0, 0/0, 0/0                          x  imported : n/a         x
x       havoc : 0/0, 0/0                               x stability : 90.61%      x
x        trim : 55.56%/4, n/a                          tqqqqqqqqqqqqqqqqqqqqqqqqj
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj            [cpu000:140%]
```

AFL Setup

- Installation – AFL site quick start and docs
- Configuration – Ubuntu specific core dumps, instrumenting target with AFL compiler
- Scripts – Ease of use
- Parallel Operation – Improve performance and input coverage
- Monitoring and Interpretation – The real art

Starting the fuzzer…

```
./afl-fuzz -t 1300 -i ../input/ -o ../findings/ ~/paho.mqtt.c/build/output/samples/paho_c_pub -t 'test/topic' -f  @@
```

Parallel Operation

- Each instance only uses one core by design
- Create master and secondary instances to improve test throughput (~2x)

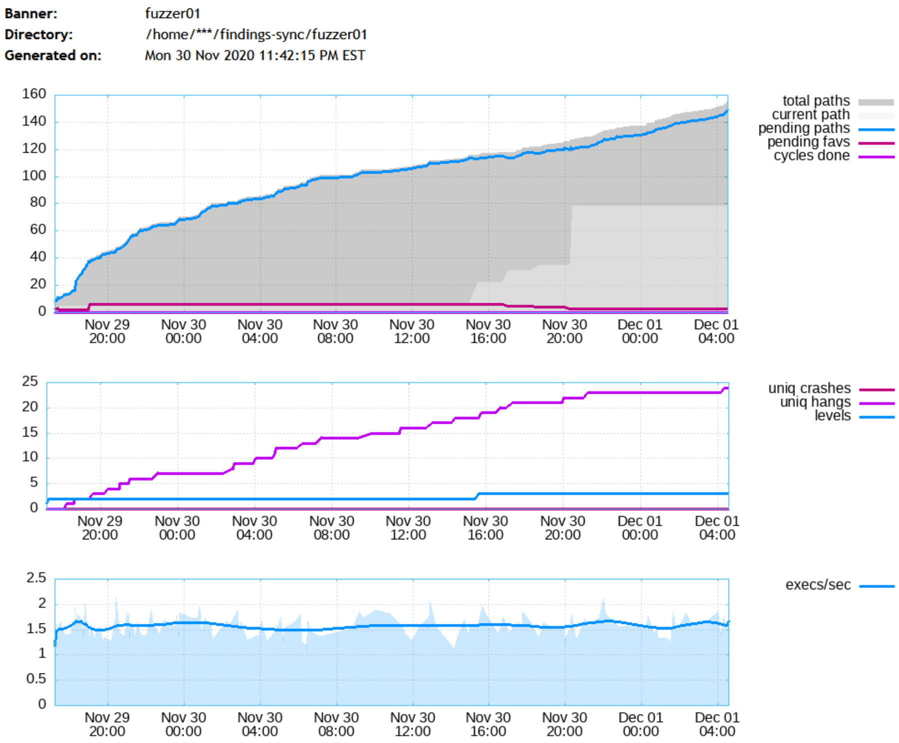
```
./afl-fuzz -t 1300 -i ../input/ -o ../findings-sync/ -M fuzzer01
         ~/paho.mqtt.c/build/output/samples/paho_c_pub -t 'test/topic' -f  @@

./afl-fuzz -t 1300 -i ../input/ -o ../findings-sync/ -S fuzzer02
         ~/paho.mqtt.c/build/output/samples/paho_c_pub -t 'test/topic' -f  @@
```

Interpretation

- Monitor AFL as it’s operating
- Use afl-plot to see overall progress
- Check hangs and crashes throughout or upon completion with AFL fuzzer stats
- Fuzzer01: 17 unique hangs
- Fuzzer02: 19 unique hangs

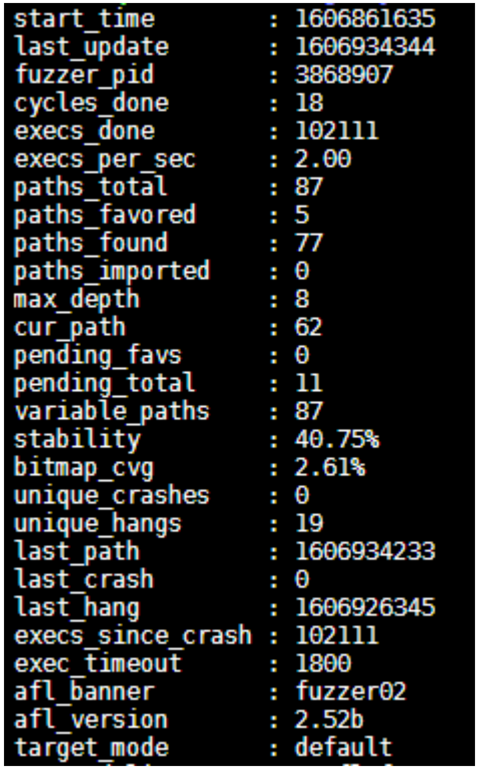
Fuzzer01: 0.00020078 hangs/exec
Fuzzer02: 0.00018607 hangs/exec



Banner:	fuzzer01
Directory:	/home/***/findings-sync/fuzzer01
Generated on:	Mon 30 Nov 2020 11:42:15 PM EST

Results

- A handful of generated inputs caused hangs
  - Execution longer than given timeout value, t (1300ms/1800ms)
- Many of these hangs were in fact valid execution of the protocol
  - MQTT specification requires the protocol to drop the connection on NUL char
- No definite vulnerabilities found, yet
- Inputs require further analysis to verify cause



```
start_time        : 1606861635
last_update       : 1606934344
fuzzer_pid        : 3868907
cycles_done       : 18
execs_done        : 102111
execs_per_sec     : 2.00
paths_total       : 87
paths_favored     : 5
paths_found       : 77
paths_imported    : 0
max_depth         : 8
cur_path          : 62
pending_favs      : 0
pending_total     : 11
variable_paths    : 87
stability         : 40.75%
bitmap_cvg        : 2.61%
unique_crashes    : 0
unique_hangs      : 19
last_path         : 1606934233
last_crash        : 0
last_hang         : 1606926345
execs_since_crash : 102111
exec_timeout      : 1800
afl_banner        : fuzzer02
afl_version       : 2.52b
target_mode       : default
```

fuzzer_stats for fuzzer02

Lessons Learned

- Fuzzing is very resource intensive (confirmed by [3])
- Fuzzing network protocols adds another layer of latency and complexity
- Take advantage of parallel capabilities

Conclusions

- Fuzzing will not always find something
- Suggests target software has achieved a baseline of stability
- Vulnerabilities could still be present
- Continue to use the same methods on more open source projects
- Interesting inputs could be forwarded to the developers of tested software

American Fuzzy Lop / wikipedia

References

1. Y. Zheng, A. Davanian, H. Yin, C. Song, H. Zhu, and L. Sun. FIRM-AFL: High-Throughput Greybox Fuzzing of IoT Firmware via Augmented Process Emulation. 2019.
2. S.H. Ramos, M.T. Villalba, and R. Lacuesta. MQTT Security: A Novel Fuzzing Approach. Wireless Communications and Mobile Computing. 2018.
3. J. Liang, M. Wang, Y. Chen, Y. Jiang, and R. Zhang. Fuzz Testing in Practice: Obstacles and Solutions. SANER. 2018.
4. Use Cases. MQTT. https://mqtt.org/.
5. A. Helin. Radamsa: A General Purpose Fuzzer. https://gitlab.com/akihe/radamsa.
6. Simon IoT. The Rise of IoT: The History of the Internet of Things. https://www.simoniot.com/history-of-iot/. 2020.
7. J. Fruhlinger. The Mirai botnet explained. CSO Online. IDG Communications, Inc. https://www.csoonline.com/article/3258748/the-mirai-botnet-explained-how-teen-scammers-and-cctv-cameras-almost-brought-down-the-internet.html. 9 Mar 2018.
8. M. Zalewski. AFL-Fuzz. https://lcamtuf.coredump.cx/afl/.

Security in Virtualized System

CS559-Final Project

Zijuan Liu

Topic

- Introduction
- Security of Hypervisor
- Security of Virtual Machine (VM)
- Security of Virtual Network

Motivation

More and more virtulized systems are **rising**, and and Google Drive is the most common for us.

Care about the security of the Google Drive, so I do a thorough survey about security of the virtualized system, and main focus on the security issues.

Introduction

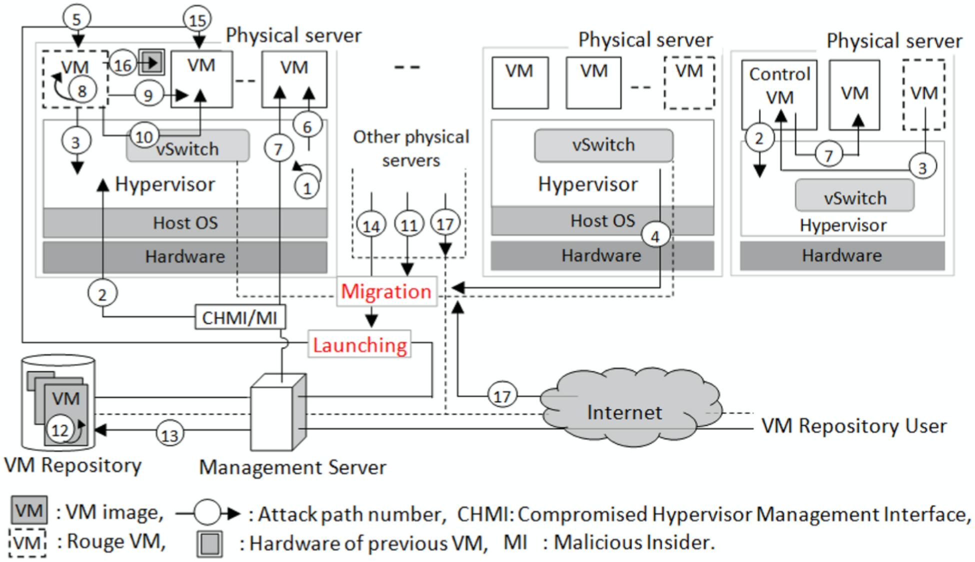
Virtualized system is an abstraction of hardware and software resources allowing heterogeneous architectures to run on the same hardware.

virtualized system includes the following components:

- **Hypervisor**
- **Virtual machine**
- **Virtual networks**
- Host OS
- Underlying hardware

One of the most popular virtualized systems

- **Cloud computing** —— top 11 threats



Related Works

Virtualized System Architecture:

”An Exhaustive Survey on Security Concerns and Solutions at Different Components of Virtualization” -- Rajendra Patil & Chirag Modi (ACM Computing Surveys, 2019)

Cloud Computing:

”Top Threats to Cloud Computing: The Egregious 11” -- Cloud Security Alliance (CSA), 2020

Security of Hypervisor

Vulnerabilities -- Causing hypervisor attack

- Uncontrolled flexibility to create VMs
- Misconfiguration
- Bugs or poor design
- Weak control over privileged and management interface
- Uncontrolled resource allocation to VM

Class of vulnerabilities

- Denial of Service (DoS)
- Gain Privilege (GP)
- Gain Information (GI)
- Code Execution (CE)

Threats -- caused by vulnerabilities

- Uncontrolled growth of VMs
- Insertion of malware / rootkits
- Unauthorized access to hypervisor resources
- Management interface compromise
- Denial of service through resource starvation by VM

Security of Hypervisor

Attacks -- serious impact on virtualization security

- Hyperjacking through VM-based rootkit (VMBR) -- Taking control over a hypervisor
- Attacks from the comprised management interface / malicious insider
- Attacks from the VM
- Attacks from the malicious hypervisor
- Launching rouge VM

Security of Virtual Machine

Security of virtual machine has 3 states

- Security of VM in running state
- Security of VM in moving state
- Security of VM in inactive state

Security of VM in Running State

Vulnerabilities

- Poor isolation between VM and hypervisor
- Poor access control over management interface
- Default state of new VMs
- Poor isolation for shared resources
- Network vulnerabilities

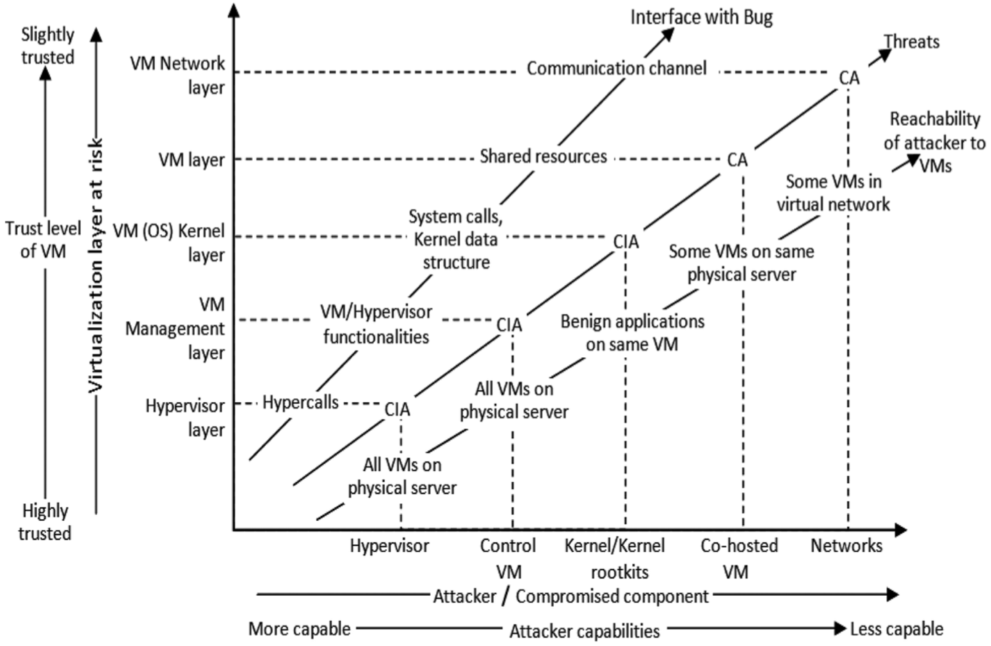
Threats

- Rootkit insertion in a VM
- Illegal access from the hypervisor management interface or a malicious insider
- Threats from the rouge VM
- Isolation failure among the VMs
- Network threats

Security of VM in Running State

Attacks

- Attacks from the compromised hypervisor
- Attacks from the compromised management interface
- Kernel-level attack -- infected VM images, allow viruses, rootkits, and other malware to do damage to a VM
- Illegal access from the co-hosted VMs
- Classical network attacks



Security of VM in Moving State

Migration of VM plays an important roles in load balancing, hardware maintenance, so it is also a obvious target for attackers.

Migration of VM could be attack by network sniffing, and malicious code injection. Besides, some attackers prefer to place themselves in the migration transit path, and then they can perform MITM attack.

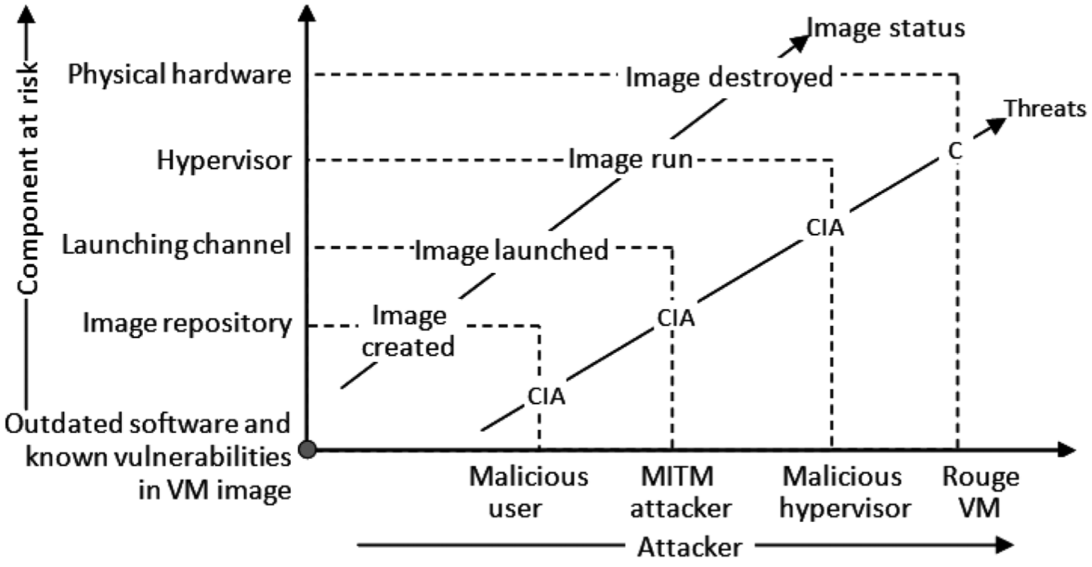
Security of VM in Inactive State

Vulnerability

- Weak access control
- Insecure launching channel
- Untrusted hypervisor

Threats

- Uncontrolled upload, creation, modification, and usage of VM images
- Unauthorized access to a launching channel and a physical device
- Deployment of the image to an untrusted hypervisor



Security of VM in Inactive State

Attacks

- Attacks on VM image contents
- Attacks on a VM image in repository
- MITM attack on VM image
- Attack on VM image at destination hypervisor
- VM data remanence attack

Security of Virtual Network

Share mode of network infrastructure increase the vulnerabilities

- DNS servers
- DHCP
- IP
- ARP protocols
- vSwitch software bugs
- Open ports
- Insecure network channels

Network attacks -- All of the network attacks are caused by the network vulnerabilities

- Denial of Service (DoS)
- Port scanning
- Sniffing
- IP / MAC spoofing

Security of Cloud Computing

According to Cloud Security Alliance (CSA), “Top Threats to Cloud Computing: The Egregious 11.” 2020.

- Data breaches
- Misconfiguration and inadequate change control
- Lack of cloud security architecture and strategy
- Insufficient identity
- Credential
- Access and key management
- Account hijacking,
- Insider threat
- Insecure interface and APIs
- Weak control plane
- Metastructure and applistructure failures
- Limited cloud usage visibility
- Abuse and nefarious use of cloud services

Conclusion

- Security issues have been discussed
- Solution for these issues are not introduced
  - Related papers in reference
- Virtualized systems no deadly security issues
  - Secure to use for **Convenience**
  - **No** Important / Sensitive information

Reference

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. ”Xen and the art of virtualization. In Proceeedings of the ACM SIGOPS Operating Systems Review”, Vol. 37, pp. 164–177, 2003.

[2] R. Patil, and C. Modi. ”An Exhaustive Survey on Security Concerns and Solutions at Different Components of Virtualization”, ACM Computing Surveys. Vol. 52. No. 1, Article 12, February, 2019.

[3] S. Singh, Y. Jeong, and J. H. Park, “A survey on cloud computing security: Issues, threats, and solutions,” Journal of Network and Computer Applications, vol. 75, pp. 200–222, 2016.

[4] Modi, Chirag, Patel, Dhiren, Borisaniya, Bhavesh, Patel, Avi, Rajarajan, Muttukrishnan, ”A survey on security issues and solutions at different layers of cloud computing”. J. Supercomput. vol. 63 (2), pp. 561–592, 2013.

[5] S. Singh, Y. Jeong, and J. H. Park, “A survey on cloud computing secu- rity: Issues, threats, and solutions,” Journal of Network and Computer Applications, vol. 75, pp. 200–222, 2016.

[6] D. Zissis, D. Lekkas, ”Addressing cloud computing security issues”. Future General Computer System, vol. 28 (3), pp. 583–592, 2012.

[7] S. Subashini, V. Kavitha, ”A survey on security issues in service delivery models of cloud computing”. J. Netw. Comput. Appl. vol. 34 (1), pp. 1–11, 2011.

[8] S. T. King and P. M. Chen. ”SubVirt: Implementing malware with virtual machines.” Proceeedings of the IEEE Symposium on Security and Privacy. pp.1–14. 2006.

[9] A. Desnos, É. Filiol, and I. Lefou. ”Detecting (and creating!) a HVM rootkit (aka BluePill-like)”. Computer Virol. vol. 7, pp. 23–49. February 2011.

[10] P. Colp, M. Nanavati, J. Zhu, W. Aiello, G. Coker, T. Deegan, P. Loscocco, and A. Warfield. ”Breaking up is hard to do: Security and functionality in a commodity hypervisor.” In Proceedings of the 23rd ACM Symposium on Operating Systems Principles. pp. 189–202. 2011.

[11] L. Shi, Y. Wu, Y. Xia, N. Dautenhahn, H. Chen, B. Zang, H. Guan, and J. Li. ”Deconstructing xen.” In Proceedings of the Network and Distributed System Security Symposium. pp. 1–15. 2017.

[12] A. Jasti, P. Shah, R. Nagaraj, and R. Pendse. ”Security in multi-tenancy cloud.” In Proceedings of the IEEE International Carnahan Conference on Security Technology. pp. 35–41. 2010.

[13] M. Kandias, N. Virvilis, and D. Gritzalis. ”The insider threat in cloud computing.” In Proceeedings of the International Workshop on Critical Information Infrastructures Security. pp. 93–103. 2011.

[14] F. Rocha and M. Correia. ”Lucy in the sky without diamonds: Stealing confidential data in the cloud.” In Proceeedings of the IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W’11). pp. 129–134. 2011.

Reference

[15] Y. Xia, Y. Liu, H. Chen, and B. Zang. ”Defending against vm rollback attack.” Proceeedings of the 42nd IEEE International Conference on Dependable Systems and Networks Workshops. pp. 1–5. 2012.

[16] S. Shafieian, M. Zulkernine, and A. Haque. ”Attacks in public clouds: Can they hinder the rise of the cloud?” Cloud Computing. pp. 3–22. 2014.

[17] J. Butler. ”Dkom (direct kernel object manipulation).” Black Hat Win- dows Security (2004). 2004.

[18] S. Bahram, X. Jiang, Z. Wang, M. Grace, J. Li, D. Srinivasan, J. Rhee, and D. Xu. ”Dksm: Subverting virtual machine introspection for fun and profit.” Proceeedings of the 29th IEEE Symposium on Reliable Distributed Systems. pp. 82–91. 2010.

[19] S. Checkoway, L. Davi, A. Dmitrienko, A. Sadeghi, H. Shacham, and M. Winandy. ”Return-oriented programming without returns.” Proceeed- ings of the 17th ACM Conference on Computer and Communications Security. pp. 559–572. 2010.

[20] M. Seaborn and T. Dullien. ”Exploiting the DRAM rowhammer bug to gain kernel privileges.” Black Hat (2015). https://www.blackhat.com/docs/us-15/materials/us-15-Seaborn- Exploiting-The-DRAM- Rowhammer- Bug- To- Gain- Kernel- Privileges.pdf . 2015.

[21] K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida, and H. Bos. ”Flip feng shui: Hammering a needle in the software stack.” Proceeedings of the USENIX Security Symposium. pp. 1–18. 2016.

[22] K. Hashizume, N. Yoshioka, and E. B. Fernandez. ”Three misuse patterns for cloud computing.” IGI Global, Pennsylvania, United States, pp. 36–53. 2013.

[23] J.Wei,X.Zhang,G.Ammons,V.Bala,andP.Ning.”Managingsecurity of virtual machine images in a cloud environment.” Proceeedings of the ACM Workshop on Cloud Computing Security. pp. 91–96. 2009.

[24] B. Grobauer, T. Walloschek, and E. Stocker. ”Understanding cloud computing vulnerabilities.” IEEE Security Privacy. Vol. 9. pp. 50–57. February, 2011.

[25] A. Pandey and S. Srivastava. ”An approach for virtual machine image security.” Proceeedings of the International Conference on Signal Prop- agation and Computer Technology. pp. 616–623. 2014.

[26] D. Reimer, A. Thomas, G. Ammons, T. Mummert, B. Alpern, and V. Bala. ”Opening black boxes: Using semantic information to combat vir- tual machine image sprawl.” Proceeedings of the 4th ACM International Conference on Virtual Execution Environments. pp. 111–120. 2008.

[27] B. Albelooshi, K. Salah, T. Martin, and E. Damiani. ”Experimental proof: Data remanence in cloud VMs.” Proceeedings of the International Conference on Cloud Computing. pp. 1017–1020. 2015.

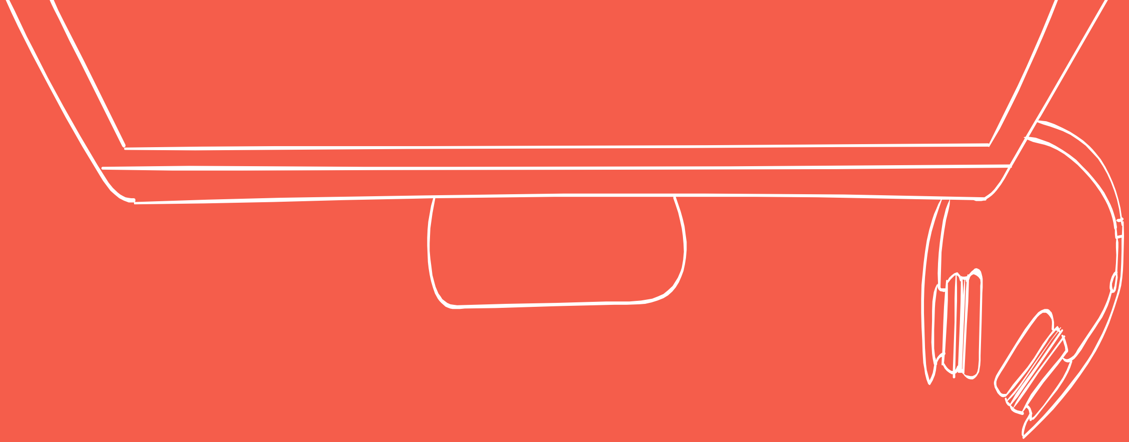
[28] R. Jain and S. Paul. ”Network virtualization and software defined net- working for cloud computing: A survey.” IEEE Community Magazine. Vol. 51, pp. 24–31. November, 2013.

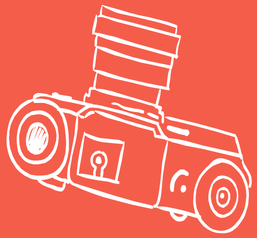
[29] CSA, ”Top Threats to Cloud Computing: The Egregious 11.” Cloud Security Alliance (CSA). April, 2020.

End~~
Thanks for Listening

Leave Questions on Discussion

Zijuan Liu Presented
12/3/2020

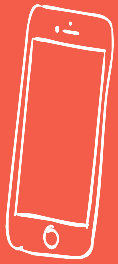


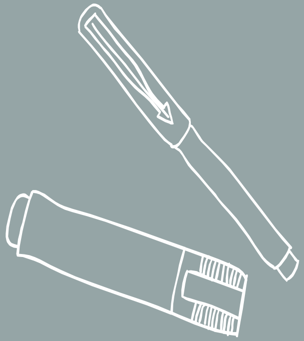


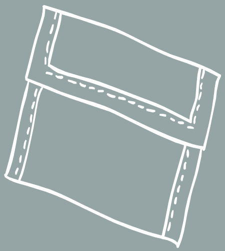
Assessing effectiveness of Penetration Testing approaches

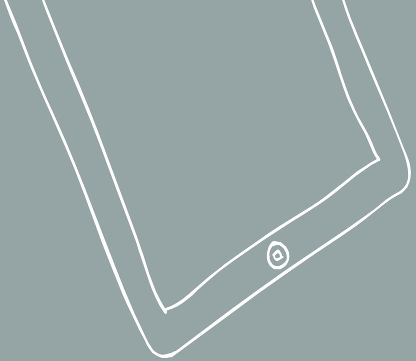
By - Siddhi Kotian

CS559 - CSU



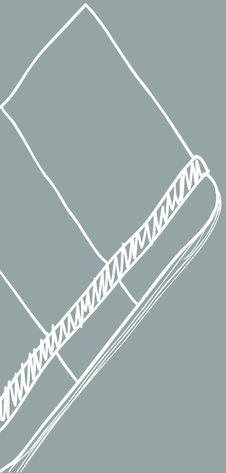


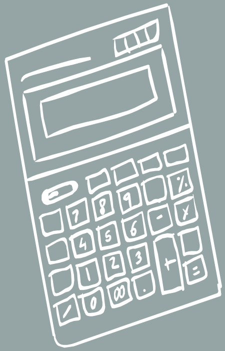




1. Penetration Testing

What is penetration testing and its types





Penetration Testing

find and exploit vulnerabilities

Average cost of data breach - $3.86

Types of Penetration testing:

- Network
- Web Application
- Wireless Network
- Social Engineering

Web Application Penetration Testing

Increase in usage of Web Application

Simulating unauthorized attacks

Finding vulnerabilities

2.
OWASP Top Ten

What are OWASP Top 10 Web Application Security Risk

OWASP Top Ten

Injection - untrusted data is sent to an interpreter
Broken Authentication - authentication system implemented incorrectly
Sensitive Data Exposure - Sensitive data not properly protected
XML External Entities (XXE) - evaluate external entity references within XML documents
Broken Access Control - what authenticated users are allowed to do are often not properly enforced

OWASP Top Ten

Security Misconfiguration - result of insecure configurations

Cross-Site Scripting XSS - application includes untrusted data in a new web page without proper validation

Insecure Deserialization - leads to remote code execution

Using Components with Known Vulnerabilities

Insufficient Logging & Monitoring

3.
web application security scanner

OWASP Zed Attack Proxy (ZAP)

Opensource

GUI based application

To access vulnerabilities in web application

Supports Scripting, Spidering and Proxying

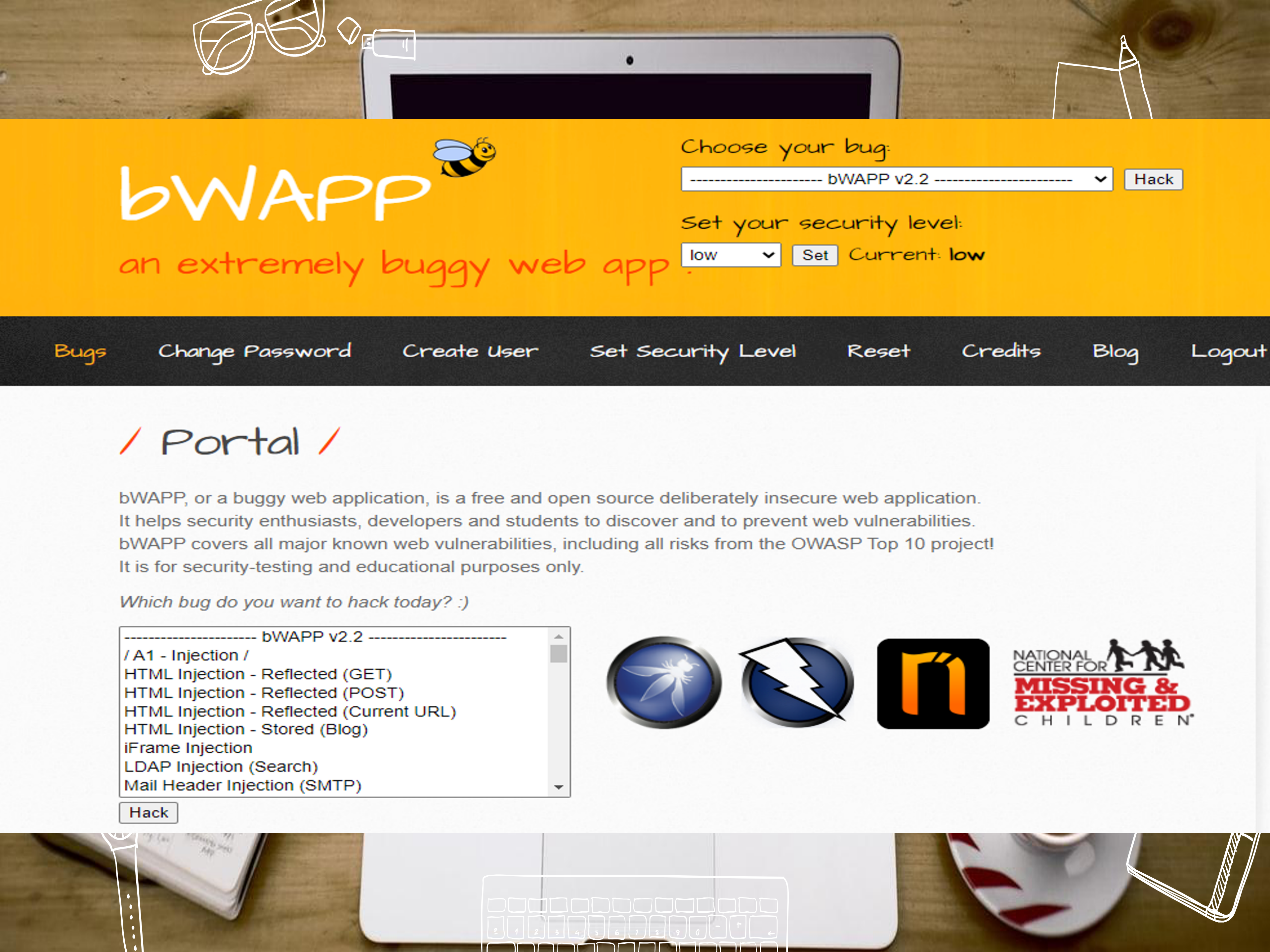
Nikto

- OpenSource
- Scans For 6,700 potential dangerous files
- Checks For Outdated Software version

4.
Comparing ZAP & Nikto

Buggy Web Application (bWAPP)

Insecure Web Application
Used for Penetration Testing
PHP as backend & MySQL Database



/ Portal /

bWAPP, or a buggy web application, is a free and open source deliberately insecure web application.
It helps security enthusiasts, developers and students to discover and to prevent web vulnerabilities.
bWAPP covers all major known web vulnerabilities, including all risks from the OWASP Top 10 project!
It is for security-testing and educational purposes only.

Which bug do you want to hack today? :)

------------------------ bWAPP v2.2 ------------------------
/ A1 - Injection /
HTML Injection - Reflected (GET)
HTML Injection - Reflected (POST)
HTML Injection - Reflected (Current URL)
HTML Injection - Stored (Blog)
iFrame Injection
LDAP Injection (Search)
Mail Header Injection (SMTP)
Hack
NATIONAL CENTER FOR MISSING & EXPLOITED CHILDREN

5.
Before Bypassing Login Page

Before Adding Cookies Or Authentication

	URLs Scanned	Time To Scan	Vulnerabilities Found
ZAP	1,497	10	9
Nikto	8,890	21 sec	7

5.

After Bypassing Login Page

After Adding Cookies Or Authentication

	URLs Scanned	Time To Scan	Vulnerabilities Found
ZAP	17,992	15 minutes	66
Nikto (same as before)	8,890	21 sec	7

Comparison

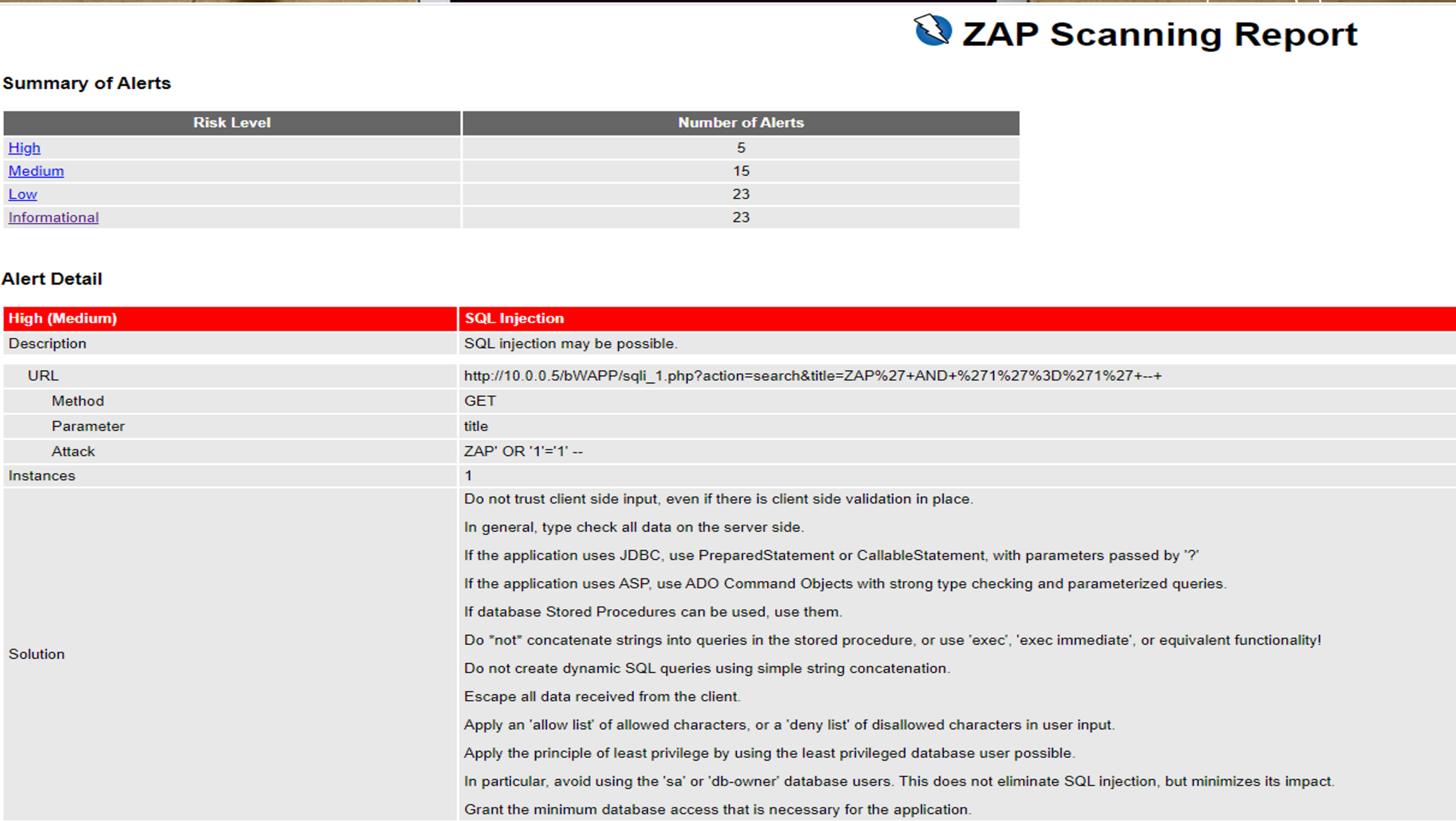
ZAP Out Performed Nikto

Nikto did not performed well after bypassing authentication

5 of the top 10 OWASP vulnerability were found

Nikto takes into the account about the version of the software used, which is helpful in initial scanning

ZAP gives the method to break into the application and what to do to fix it



ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	5
Medium	15
Low	23
Informational	23

Alert Detail

High (Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://10.0.0.5/bWAPP/sqli_1.php?action=search&title=ZAP%27+AND+%271%27%3D%271%27+--+
Method	GET
Parameter	title
Attack	ZAP' OR '1'='1' --
Instances	1
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply an 'allow list' of allowed characters, or a 'deny list' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.

Thanks!

Any questions?