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
- Learn 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 Card username and password
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
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:**
  - \[ \text{Risk} = (\text{MR}*I) + (\text{UR}*I) + (\text{TR}*I) + (\text{VR}*I) \]
## Risk Assessment Model

### Risk Assessment

<table>
<thead>
<tr>
<th>System</th>
<th>MR</th>
<th>UR</th>
<th>TR</th>
<th>VR</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DS1</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>3DS2</td>
<td>Very Low</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Apple Pay</td>
<td>Very Low</td>
<td>Very Low</td>
<td>Very Low</td>
<td>Very Low</td>
</tr>
</tbody>
</table>

### Apply Risk Formula

<table>
<thead>
<tr>
<th>System</th>
<th>MR</th>
<th>UR</th>
<th>TR</th>
<th>VR</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DS1</td>
<td>1.5</td>
<td>2.4</td>
<td>1</td>
<td>1.6</td>
<td>6.5</td>
</tr>
<tr>
<td>3DS2</td>
<td>.3</td>
<td>1.5</td>
<td>.2</td>
<td>.2</td>
<td>2.2</td>
</tr>
<tr>
<td>Apple Pay</td>
<td>.3</td>
<td>.3</td>
<td>.2</td>
<td>.2</td>
<td>1</td>
</tr>
</tbody>
</table>

### 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 Card 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
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
28

- Use of insecure data transfer protocols: 83% (2017) to 81% (2018)
- Dictionary passwords: 75% (2017) to 71% (2018)
- Vulnerable software versions: 67% (2017) to 75% (2018)
- Interfaces for remote access, hardware management, and DBMS connections are available to any Internet user: 58% (2017) to 69% (2018)
- Storage of sensitive data in cleartext or available to the public: 58% (2017) to 56% (2018)
- Excessive application or DBMS privileges: 42% (2017) to 25% (2018)
- No authentication for access to critical resources: 25% (2017) to 19% (2018)
- SQL Injection: 17% (2017) to 19% (2018)

High or Critical
Medium
2017
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%
- Insufficient protection against recovering credentials from OS memory: 86%
- Protocol security flaws leading to traffic redirection and interception of network configuration details: 100%
- Insufficient protection of privileged accounts: 56%
- Storage of sensitive data in cleartext: 44%
- Excessive application or DBMS privileges: 22%
- Vulnerable software versions: 44%
- Arbitrary File Reading: 33%
- SQL Injection: 22%
- Remote Code Execution: 22%

Legend:
- High or Critical
- Medium
- 2017
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

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

**Nmap**
Network Mapper is a network discovery and security auditing tool

**Masscan**
Masscan is a internet port scanner
## Nmap vs masscan

<table>
<thead>
<tr>
<th></th>
<th>Time Taken</th>
<th>CPU Utilization</th>
<th>Scans TCP and UDP protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nmap</td>
<td>11.3</td>
<td>0%</td>
<td>yes</td>
</tr>
<tr>
<td>Masscan</td>
<td>4.06</td>
<td>2%</td>
<td>yes</td>
</tr>
</tbody>
</table>
### Sqlmap vs DSSS

<table>
<thead>
<tr>
<th></th>
<th>Time Taken</th>
<th>CPU Utilization</th>
<th>Successful Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>SqlMap</td>
<td>19 s</td>
<td>11%</td>
<td>yes</td>
</tr>
<tr>
<td>DSSS</td>
<td>2.9 s</td>
<td>6%</td>
<td>no</td>
</tr>
</tbody>
</table>
THANKS!

Any questions?
References


Fuzzing Open Source IoT Projects

Brett Mulligan
CS559
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]

Motivation: MQTT

- Message Queuing Telemetry Transport
- 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
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
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
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
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
References


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 virtualized systems are rising, 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

- 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


Reference


End~~
Thanks for Listening

Leave Questions on Discussion
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
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
Want big impact
Use big image.

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? :)

Choose your bug:

Set your security level:
low  Set  Current: low
5.
Before Bypassing Login Page
### Before Adding Cookies Or Authentication

<table>
<thead>
<tr>
<th>Tool</th>
<th>URLs Scanned</th>
<th>Time To Scan</th>
<th>Vulnerabilities Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZAP</td>
<td>1,497</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>Nikto</td>
<td>8,890</td>
<td>21 sec</td>
<td>7</td>
</tr>
</tbody>
</table>
5. After Bypassing Login Page
## After Adding Cookies Or Authentication

<table>
<thead>
<tr>
<th>Tool</th>
<th>URLs Scanned</th>
<th>Time To Scan</th>
<th>Vulnerabilities Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZAP</td>
<td>17,992</td>
<td>15 minutes</td>
<td>66</td>
</tr>
<tr>
<td>Nikto (same as before)</td>
<td>8,890</td>
<td>21 sec</td>
<td>7</td>
</tr>
</tbody>
</table>
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
Summary of Alerts

<table>
<thead>
<tr>
<th>Risk Level</th>
<th>Number of Alerts</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>5</td>
</tr>
<tr>
<td>Medium</td>
<td>15</td>
</tr>
<tr>
<td>Low</td>
<td>23</td>
</tr>
<tr>
<td>Informational</td>
<td>23</td>
</tr>
</tbody>
</table>

Alert Detail

<table>
<thead>
<tr>
<th>Description</th>
<th>SQL Injection</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL injection may be possible.</td>
<td></td>
</tr>
</tbody>
</table>

**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?