Homework 3
CS 468: Network Security

October 13, 2023

Getting Started

The homework assignment due date is **October 27, 2023 at 11:59pm local time**. For this assignment, you are required to write your solution in Python programming language, version 3.5 or later. The goal of this assignment is to write an on-path DNS packet injector To complete this assignment, you are allowed to use standard Python libraries as well as packet generating modules such as Scapy, dpkt etc. All files required for this assignment are provided along in `hw3_files.zip`

Task 1: Writing a DNS Injector (10 Points)

In class we talked about DNS poisoning from different vantage points. One such vantage point is **on-path**, where an attacker can passively monitor traffic which include DNS queries and can inject properly forged responses. Such attacks can be carried out by an adversary observing traffic on open WiFi networks, or malicious ISPs. Depending on the origin of the packets (which can be a DNS resolver or an end machine) an attacker can poison the resolver cache, or send the client an incorrect DNS resolution.

In this assignment your goal is to write an on-path DNS injector which will capture traffic from a network interface in promiscuous mode, and attempt to inject forged responses to selected DNS A requests. You should write your program in a single file and name it as `dnsinject.py`.

Your DNS injector should execute using the following command and take in the following inline arguments:

```
>> python dnsinject.py [-i interface] [-h hostnames]
```

- `-i`: Listen on network device interface (e.g., `eth0`). If not specified, your program should select a default interface to listen on. The same interface should be used for injecting forged packets.
- `-h`: Read a hostname file containing a list of IP address and hostname pairs specifying the hostnames to be hijacked. If '-h' is not specified, your injector should forge replies for all observed requests with the local machine's IP address as an answer.

Hints

- To complete this assignment, you will need to understand the format of a DNS query and response. Here is a resource that you might find useful.
- As the homework requires you to observe DNS packets, to make your code efficient you can apply a filters for the packets you intercept. This ensures you are looking at only the relevant packets.
- The Wireshark utility can be used to capture packets and investigate the values. This will be helpful when you are crafting your packet and comparing it with a legitimate DNS response.

- Pay attention to the header values in the request and the response. Some will remain the same while the others will be flipped. You will also have to add the IP in the response from the hostname file (if provided).
- DNS uses the TXID and source port randomization to make guessing hard. With the vantage point of an on-path attacker you are able to see both these these.

There are multiple ways to approach this problem. Be creative in your approach! If you have any questions, use the class Piazza for discussion.

Testing Your Code

One way to test your tools is to have a guest VM running on your computer that makes DNS queries. Your DNS injection program running on the host OS and is able to view those queries, and inject forged responses. To make requests within the VM you may use the dig utility.

Your program will be tested for both correctness as well as efficiency. By correctness, we mean that your forged responses are properly formed and accepted by domain lookup tools such as dig and nslookup. While testing your code, you can force requests to non-existent DNS resolvers. In such a case a legitimate response will never arrive and if your forged response is accepted by the DNS lookup tool, you know you have crafted a successful packet.

To ensure for efficiency, make sure your program is fast enough and inject packets to the correct destination before the legitimate server response arrives. Remember, to successfully forge a response, you have to win the race condition. We will be testing your code against the default UIC resolver, as well as common open DNS resolvers.

Alternative Testing Setup: If you are unable to setup a VM, you can use the dig utility within your host system to generate DNS requests. You can then run your program to observe packets on the same interface you are making queries on, and inject forged responses.

What to submit

For this task you are required to upload the following files in an archive named as `hw3.zip` to Gradescope.

1. `dnsinject.py`: Your main program that performs DNS injection
2. `injection.pcap`: A pcap file containing the relevant packets of a successful injection attack that you have conducted. Make sure to filter out any traffic which is unrelated to your attack.
3. `explanation1.txt`: Describe in a paragraph about your approach on injection, how you successfully tested it, and list the links of online resources you referred to.

Task 2: DNS poisoning detector (10 Points)

Task A: Identifying spoofed DNS responses

For this task, you are asked to analyze the `injection.pcap` of Task1. How do legitimate DNS responses differ from spoofed ones? Which fields and headers are the same and which are modified? Write a detailed report of your analysis and clarify your observations.

Task B: Implementing the detector

Using your findings of the previous task, you are asked to implement the DNS poisoning detector. Specifically, your tool has to capture the traffic from a network interface in promiscuous mode and detect DNS poisoning attack attempts, such as those generated by your dnsinject tool or by dnssnif. Detection is based on identifying duplicate responses, which contain different answers for the same domain's request (i.e., the observation of the attacker's spoofed response and the server's actual response). The order of arrival should not matter: you should raise an alert irrespectively of whether the attacker's spoofed response arrived before or after the actual response. You should make every effort to avoid false positives, e.g., consecutive legitimate responses with different IP addresses for the same hostname due to DNS-based load balancing. Your tool should conform to the following specification:

```
>> python dnsdetect.py [-i interface] [-r tracefile ]
```

- `-i`: Listen on network device interface (e.g., `eth0`). If not specified, your program should select a default interface to listen on.
- `-r`: Read packets from `tracefile` (tcpdump format). If "-r" is not specified, your tool should detect attempts of DNS spoofing for all the requests of the local machine's IP address.

Once an attack is detected, your tool has to create a log file named: attack log.txt, containing both spoofed and legitimate responses. The log file must contain the date and time of the detected response, the DNS transaction ID, the attacked domain name, and the original and malicious IP addresses. For example:

– **October 20 2023 18:34:02**
– **TXID 0x5cce Request www.example.com**
– **Answer1 [List of legit IP addresses]**
– **Answer2 [List of malicious IP addresses]**

Please follow the above structure of the log file. If you want to add further information or additional fields, feel free to include them in your explanation file.

What to submit

For this task you are required to submit the following files :

1. `spoofed.txt`: The analysis required for Task A.
2. `dnsdetect.py`: Your main program that performs DNS spoof detection.
3. `attack_log.txt`: The output of your tool containing all the required information of the DNS spoofed responses.
4. `explanation2.txt`: Describe in a paragraph about your approach on detection, how you successfully tested it, and list the links of online resources you referred to.

Submission

To submit, create a zip archive of the 7 files, named hw3.zip and upload it to Gradescope.

(Part of the assignment is originally from Stony Brook University CSE508 taught by Mike Polychronakis in 2021. Thanks!)

Good Luck!