DNS Exfiltration

## Summary

This lab shows how an attacker can exfiltrate data using DNS requests.

It is effective because firewalls typically do not block or even monitor outbound DNS requests.

## Tools/Resources

Parrot: Python3, nslookup

## Walkthrough

1. Review code on Github: https://github.com/kengraf/DNSexfil/edit/main/dns\_exfil\_server.py
   1. What port does it run on? **53**
   2. What functions are called? **handler**
   3. Does it use any libraries? **io, re, struct, socketserver, base64**
   4. Does it call any external services? **No**
   5. What is the structure of the DNS request? **First (least significant) subdomain is base64 encoded and the only part acted upon.**
2. Download program: curl https://raw.githubusercontent.com/kengraf/DNSexfil/main/dns\_exfil\_server.py > dns-exfil.py
3. sudo python3 dns-exfil.py

A screen shot of a computer

Description automatically generated

1. Build a base64 message to send
   1. base64, add your message, Ctrl-D
   2. Ignore any trailing equal signs (=)

A screenshot of a computer

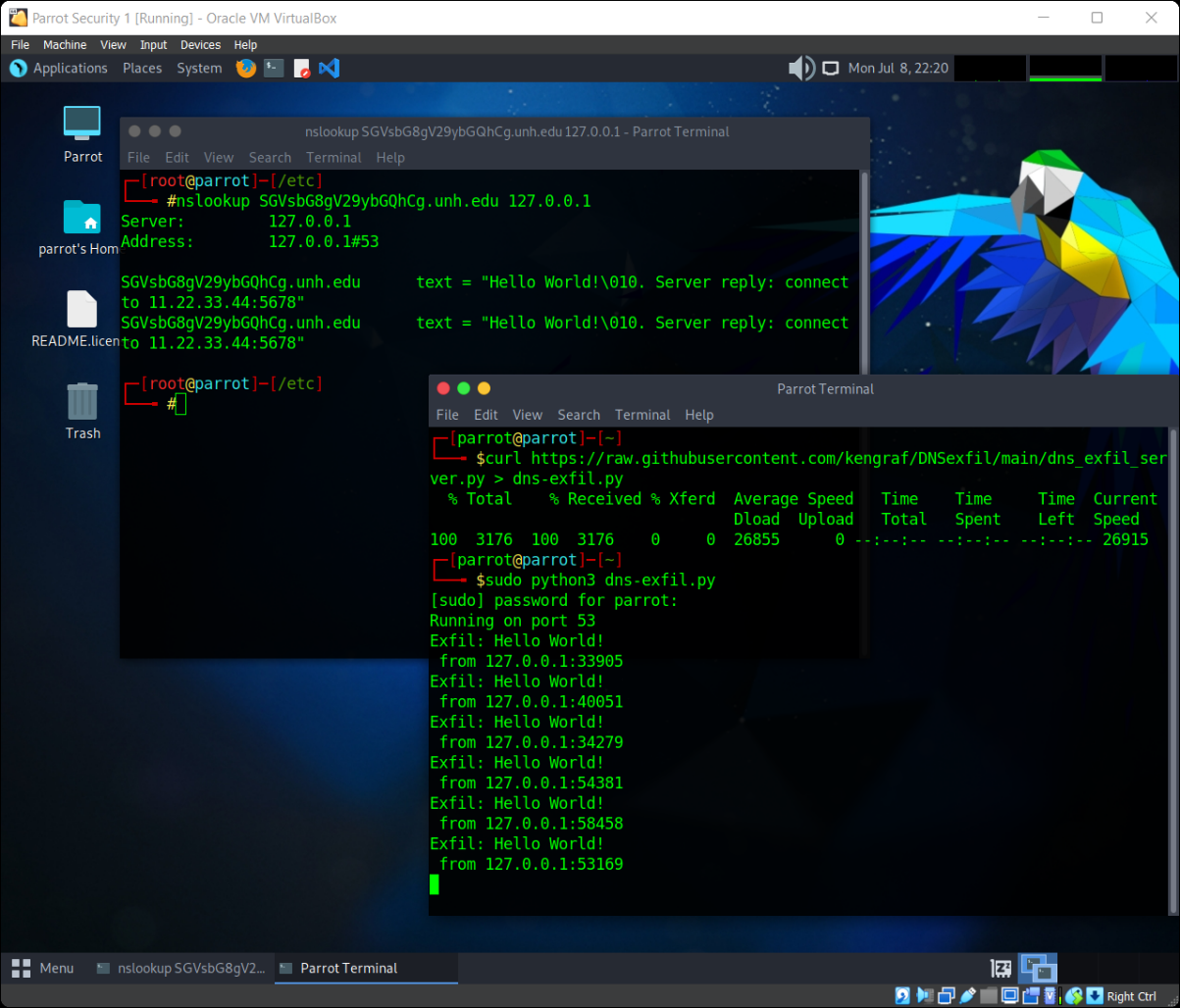
Description automatically generated

1. Does the behavior change if we:
   1. Try “unh.edu” as the base domain?
   2. Set unh.edu to localhost in /etc/hosts?

## Submission

Upload to Canvas a docx or pdf containing a screenshot showing:

1. Your DNS exfil server answering requests
2. Using nslookup to make requests to your DNS exfil server



## Additional Thoughts

How does an organization detect the exfiltration? **One-offs would slide, repeated requests maybe not**

How much data can be sent? **63 characters, base64 encoded = 378 bits**

Could BOTs use this method to communicate with each other? Yes, but requests to unofficial DNS servers might standout.

Why does nslookup generate 2 requests? **IPv4 and IPv6**

How does this attack impact orgs that run internal DNS servers? **Firewalls see the request coming from the internal DNS server not th machine that made the request.**