Firewall and DNS Filtering Penetration Tests using Parrot OS

Aparicio Carranza, PhD[1], Miguel Bustamante, PhD[2], Harrison Carranza, MSIS[2], Casimer DeCusatis, PhD[3], Jennifer Islam, BTECH[1]

[1]*The New York City College of Technology – CUNY, USA, acarranza@citytech.cuny.edu*

[2]*Vaughn College of Aeronautics and Technology, USA, miguel.bustamante@vaughn.edu, harrison.carranza@vaughn.edu*

[3]*Marist College, USA, casimer.decusatis@marist.edu*

***Abstract* – Cybersecurity threats targeting user privacy and DNS have become increasingly common, especially due to the increasing number of remote users driven by the COVID-19 pandemic. In this tutorial paper, we address configuration of a penetration testing environment for preventing information leakage that could compromise anonymous network services. We first construct and validate a test environment using the security edition of the Parrot operating system for GNU/Linux and Windows environments, and establish its resilience against various scans and attacks, including a Synchronize (SYN) scan in stealth mode with spoofed source IP address, FTP bounce scan, web host vulnerability scans using Nikto, and denial of service (DoS) attacks using Metasploit. Experimental results are documented using Nmap and WireShark. We then use this environment to test DNS leakage and transparent DNS proxy effects. Mitigation for these attacks is demonstrated using the AnonSurf application bundled with Parrot OS.**

***Keywords:* DNS Filtering, Firewall, Parrot OS, Penetration Testing**

I. Introduction

Cybersecurity threats have become increasingly common, especially in recent years due to the large increase in remote work and education caused by the COVID-19 pandemic. Attacks against high value assets such as Domain Name Services (DNS) have shown a significant increase. According to recent industry reports, 87% of organizations worldwide suffered DNS attacks in 2020, with the average attack costing around $880,000 [1]. There is a need for additional ethical penetration testing to guard against DNS attacks, combined with a defense in depth strategy including firewalls, antivirus, and anonymization technologies. In this tutorial paper, we construct a penetration test environment for these systems based on the open source Parrot operating system and related tools, and experimentally demonstrate their effectiveness against commonly employed reconnaissance scans and attack methods. Parrot OS (*Parrot Operating System*) is a free, open source GNU/Linux distribution (based on Debian Linux) with an emphasis on cybersecurity, penetration testing, and privacy [2]. It is available in several editions, including a version with standard home desktop tools which can be used as a general-purpose operating system. Parrot OS is compatible with virtual environments and cloud installations, and can run as a Docker container. It can also dual boot with Microsoft Windows. The security edition used in this paper has its own application repository, including all the packages supported by Debian as well as additional applications (such as a compatibility layer which allows Windows applications to run in GNU/Linux environments). By default, Parrot OS disables all the network services pre-installed in the system. This avoids service exposure in a target network, and provides a very low RAM footprint which has security advantages and also improves performance. Each network service needs to be manually started when needed. The operating system is designed to integrate tools used for professional penetration testing and forensics analysis, providing these programs with easy root access privileges. However, although it is comparatively easy to run security testing with root access, the system also protects against unauthorized root access exploits by using Linux hardening technologies to sandbox the operating system. Automount functions are disabled by default, so that forensic data acquisition can be performed safely. This includes file manager setting to disable plug-and-play applications, and a no-automount kernel option passed by default at boot (write blockers should still be used to guard against accidental mounts). For this paper, we ran Parrot OS on a quad core x86-64 processor with 8 GBytes DRAM and 128 GBytes SSD storage. The installation instructions, including download validation with an MD5 hash, are documented elsewhere [2] and will not be repeated here.

The remainder of this paper is organized as follows. After the introduction, we discuss setup and configuration of the penetration test environment, including the GUI for the Uncomplicated (GUFW) firewall, ClamAV antimalware, and Proton VPN. We then attempt various scans and attacks, including a SYN scan in stealth mode, FTP bounce scan, web host vulnerability scans using Nikto, and denial of service (DoS) attacks using Metasploit. Experimental results are documented using Nmap and WireShark during these tests. After establishing the baseline behavior of this environment,

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2022.1.1.675
ISBN: 978-628-95207-0-5 **ISSN:** 2414-6390

we then test DNS information leakage and transparent DNS proxy effects. Mitigation for these attacks is demonstrated using the AnonSurf application bundled with Parrot OS.

II. CYBERSECURITY PENETRATION TESTBED INSTALLATION

We created a penetration testbed for evaluation of server reconnaissance scanning, firewall penetration, resistance to common Denial of Service (DoS) attacks, and information leakage. These experiments used the Linux Parrot OS (a penetration testing operating system similar to Kali Linux) installed under an Oracle VirtualBox hypervisor. We used the widely adopted GUFW firewall application for Ubuntu Linux, which can be installed with root access privileges from their GitHub site [3]. Both introductory and advanced stateful firewall configurations can be configured from the firewall graphical user interface. Additionally, we installed antivirus and malware protection using the free ClamAV application [4]. ClamAV features include a command-line scanner and an automatic database updater with a scalable multi-threaded daemon, which runs on the antivirus engine from a shared library [4], [5]. The command line software interface is called Clamtk, as shown in Fig. 1, which allows us to configure a scheduled antivirus scan or scan on demand at any time, for either single files or entire directories.

Fig. 1 Installing the malware protection ClamAV using Clamtk.

We then installed a free virtual private network (VPN) supported under ParrotOS, the ProtonVPN [6], which allows us to configure VPNs that are valid in the U.S. and several other countries as shown in Figs. 2 and 3. The installation website can automatically generate an Internet Key Exchange (IKEv2) username and strong password for VPN authentication.

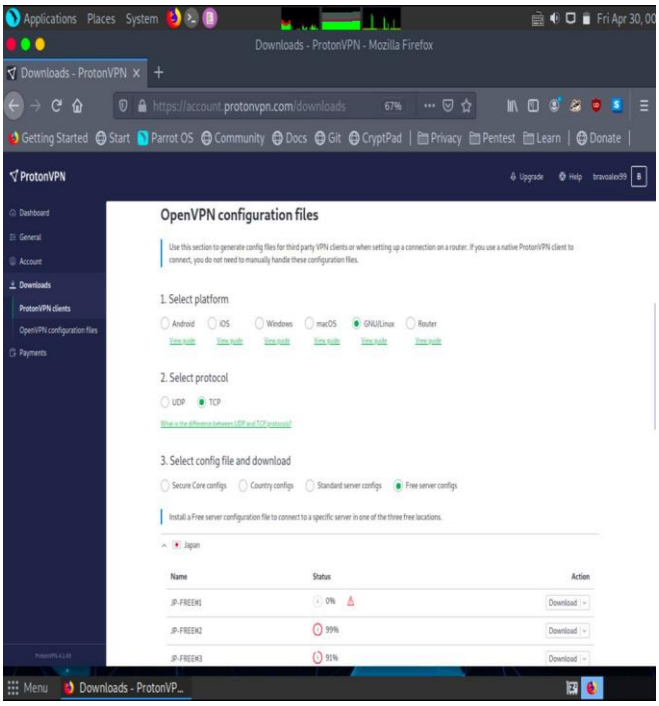

Fig. 2 Options for VPN configuration

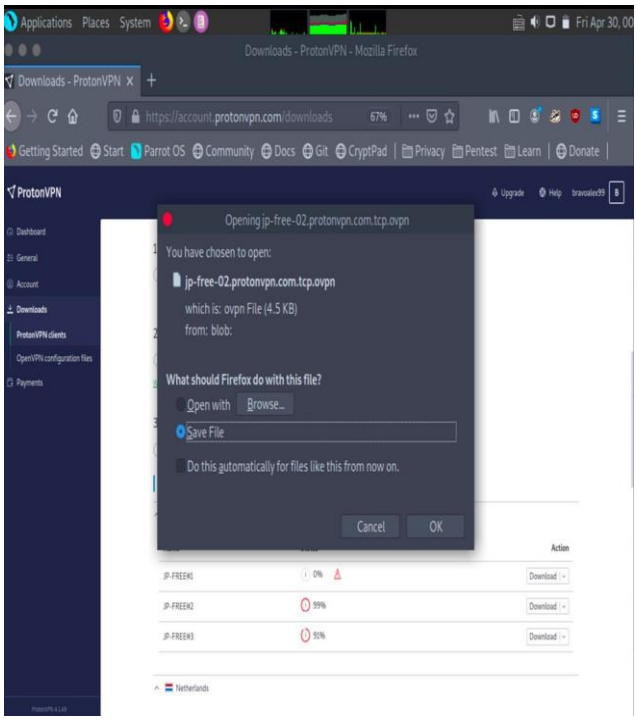

Fig. 3 Downloading the VPN unto the computer WireShark.

III. BASELINE PENETRATION TESTING

In the following sections, we will describe the specific approach used to set up, configure, and penetration test this environment; all command line prompts will be shown in bold italic text to simplify reproduction of this exact environment and corresponding test results. We verified the experimental results from this test environment using well established tools such as Nmap and WireShark. In order to penetration test the firewall, we first attempted a SYN scan in stealth mode with a decoy source IP address. This was accomplished using the Nmap command ***nmap –sS –D [IP address information]*** (where we have redacted the IP addresses used in our testing). By observing the network traffic behavior using WireShark, we observed that the decoy IP address was effectively blocked and a RST was sent in response to the SYN scan, as shown in Fig. 4.

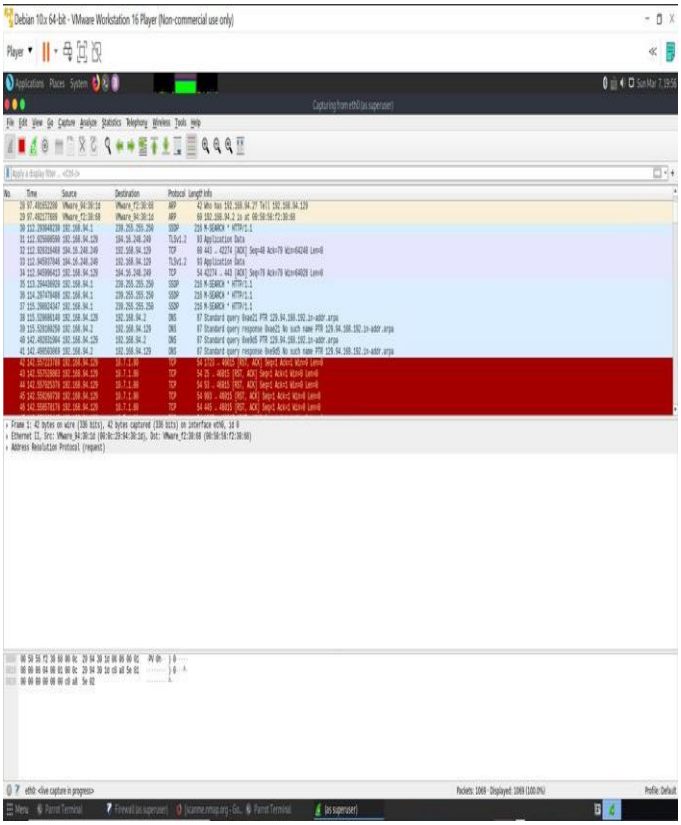

Fig. 4 Blocking communication to the decoy IP address shown in Wireshark.

Next, in an effort to bypass the firewall, we enacted an FTP bounce scan, which employs a third-party host for reflecting port scans and connection attempts against the target IP address. This scan can be implemented on Nmap using the command ***nmap –p22,25,135 –Pn –v –b [IP address] scanme.nmap.org***. Once again, the firewall was able to successfully detect the bounce scan (shown by the black highlighted field on the WireShark trace in Fig. 5) and block communication to the target IP address with a RST response packet.

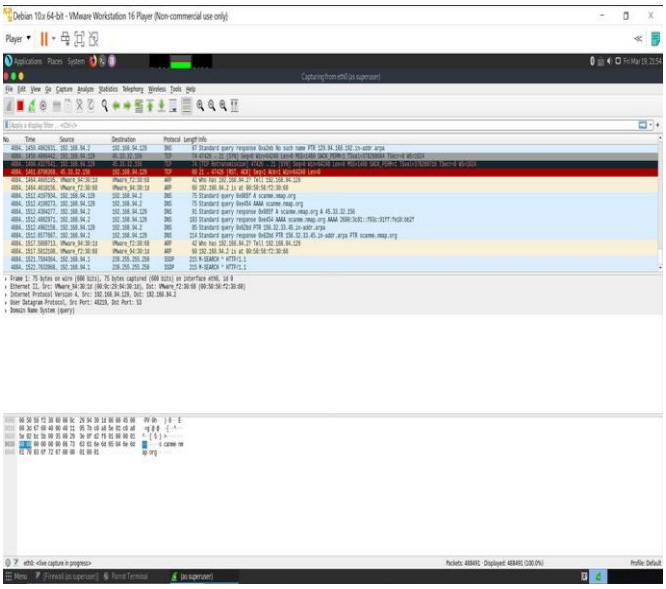

Fig. 5 Bounce Scan to target IP address.

Continuing the reconnaissance scan testing, we used the tool Nikto [7] to perform integrated scanning and testing of a website hosted on the target IP address, to see whether we could find any exploitable vulnerabilities. Nikto is an open source web server scanner for evaluating server configurations or checking the version of different application programs. Using the command ***nikto -h webscantest.com*** to scan the target web server, we identified a number of common vulnerabilities as shown in Fig. 6, although none of these related to the DNS and all have patches readily available.

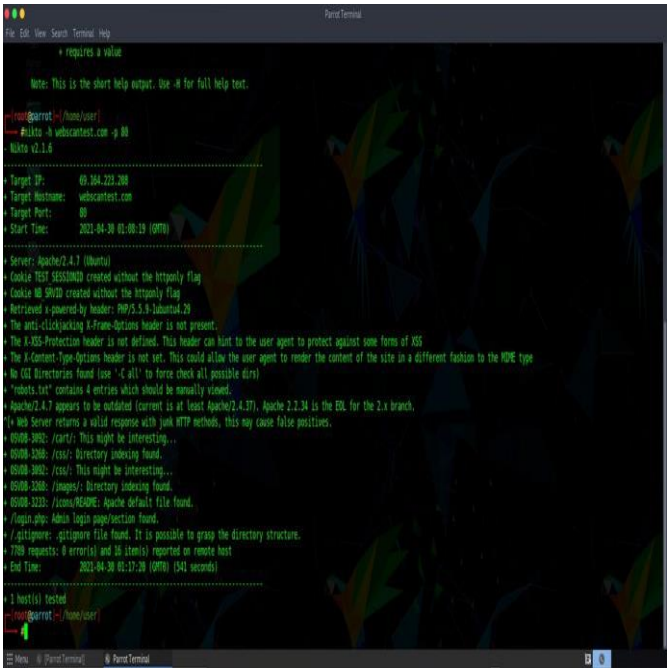

Fig. 6 Detecting the web server using Nikto.

The next step in validating our test environment involved attempting a series of simulated DoS attacks. First, we performed a simulated DoS attack against the target using an Nmap script, which was invoked using the command ***nmap --script dos –Pn scanme.nmap.org*** as illustrated in Fig. 7. In this test, the firewall was able to block the DoS attack from flooding ports on the target system, as shown by the WireShark trace in Fig. 8.

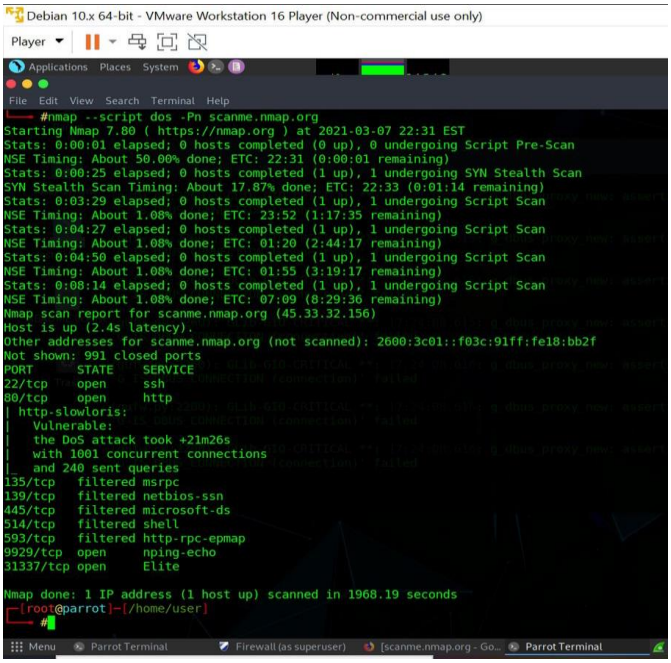

Fig. 7 DoS attack execution on Nmap.

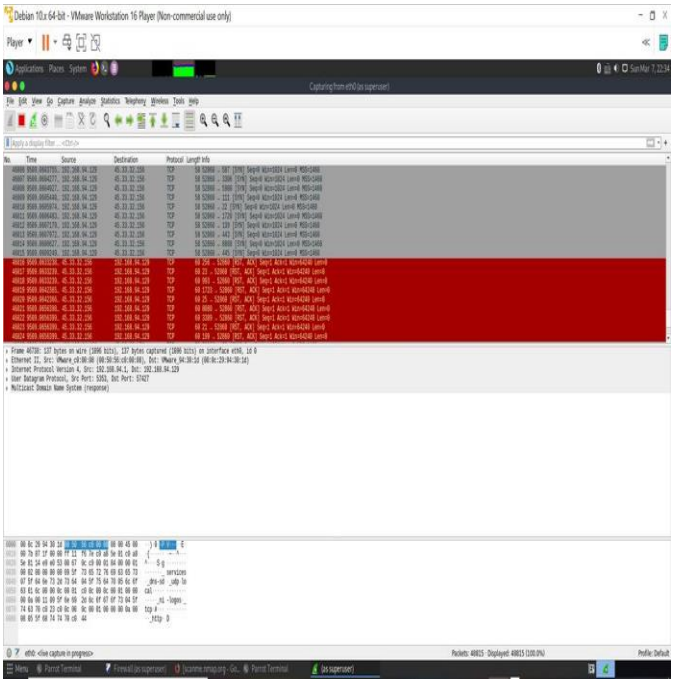

Fig. 8 Block communication to target IP address.

Then we attempted another form of DoS attack using the Metasploit tool [8]. By running the command ***use auxiliary/dos/tcp/synflood*** we can set up a SYN flood attack from an existing library in auxiliary mode. Next, the command ***set RHOST [IP address]*** directs this attack against the target IP address, and the attack can then be executed using the command ***exploit***. Network traffic was monitored during the attack using WireShark, which proved that the attack was successfully detected and blocked as illustrated in Fig. 9.

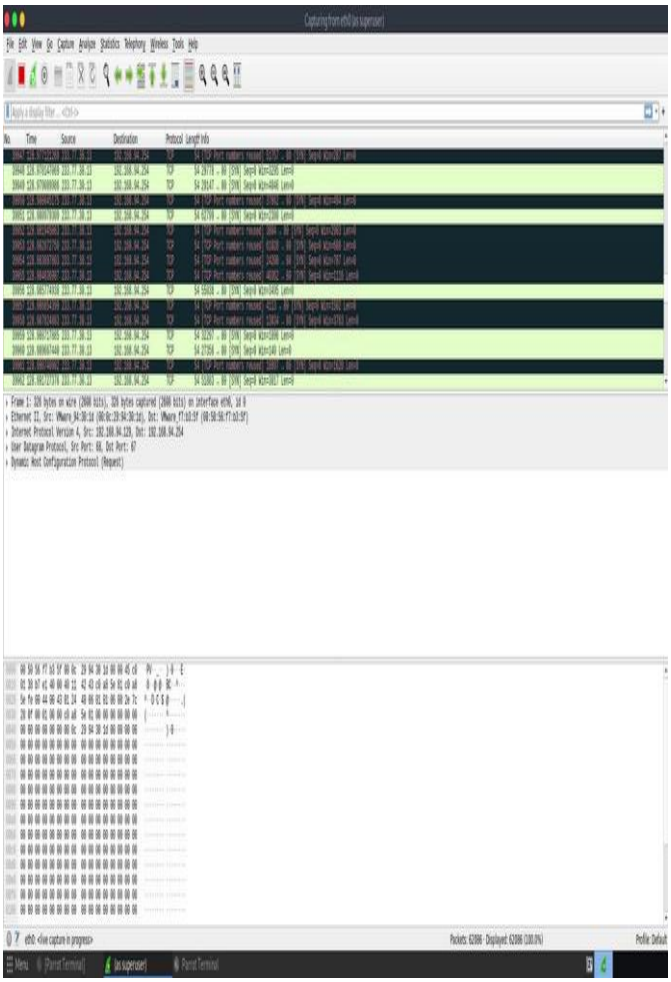

Fig. 9 Wireshark detecting and blocking the DoS attack.

These tests validated that our baseline environment was resistant to common forms of Confidentiality, Integrity, and Availability attacks including different forms of stealth scanning and DoS floods. Our next step was to evaluate DNS attacks which could compromise privacy, and recommend mitigation using tools available in Parrot OS.

IV. DNS FILTERING

Many web services implement DNS filtering techniques to enforce security policies, such as blocking access or downloads from specific countries [9]. One approach to

bypassing DNS filtering is the use of VPNs to enable anonymous access to a website. When using such a service, it is important that all traffic is routed through the anonymizing network, otherwise an adversary can monitor and filter the traffic. However, under some conditions, even when a computer is connected to an anonymizing network, the operating system will continue using its default DNS server, instead of the anonymous DNS service provided by the anonymizing network. This is known as DNS leakage, and under these conditions private data can still be viewed by an adversary despite the apparent connection to an anonymizing network. This can provide a false sense of security, which is why detecting and preventing DNS leaks is so important. To demonstrate this, we used the open source tool available from the website dnsleaktest.com on a sample VPN [10]. After verifying that the working VPN did not disclose the actual IP address or geolocation information, we ran an extended version of the test as shown in Fig. 10, which is used to determine if there are specific websites or resources not protected by a given VPN.

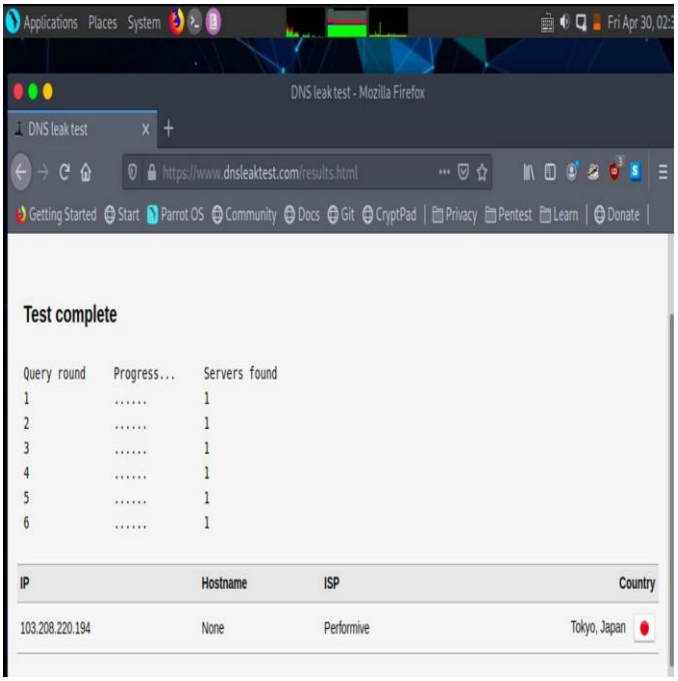

Fig. 10 Extended DNS leak test.

Some network service providers use a technique called Transparent DNS proxy, which intercepts all DNS lookup requests (from well-known port 53) and transparently proxies the results. This effectively means that a user has to use the service provider's DNS service for all DNS lookup requests. This technique is also used by so-called "open" DNS services, such as those from Google or OpenDNS. This can also be tested using the same approach described previously. Implementing a transparent DNS proxy can also result in DNS leakage, since DNS requests made while connected to the VPN may still be intercepted by a third party.

In order to remove DNS leaks, we need to ensure that only the DNS servers provided by the VPN service are being used whenever the VPN is connected. This can be prevented by using more recent versions of open DNS services, for example OpenVPN version 2.3.9 or greater allows the prevention of DNS leaks by spec. Another approach for older versions of software involves setting static IP address properties for DHCP before connecting to the VPN, removing DNS settings after connecting to the VPN, and switching back to DHCP or the original DNS server after disconnecting from the VPN. Switching to a static IP configuration prevents the DNS settings from being overwritten if your computer renews its IP address while connected to the VPN. The DNS leakage test described above can be used to verify the proper operation of an anonymizing service, and it is considered best practice to flush the DNS resolver cache after disconnecting from such a network.

Another method to bypass DNS filtering is the use of AnonSurf [11], which is provided as part of the ParrotOS package. This script routes all network traffic through TOR, and also supports anonymous peer-to-peer distributed communications (I2P services) as well as clearing any identifying information leakage from the user's computer. Any traditional network service can run over I2P, including classic web browsing and hosting as well as distributed applications such as DNS. Further, Anonsurf includes the application Pandora, which automatically overwrites RAM when the system shuts down to clean application caches and remove any information which can be traced back to the end user (so-called Pandora bombing). While Anonsurf is not legal in all countries, it can currently be used in North America. The application GUI provides a user-friendly interface which includes the ability to change a user's identity, get their IP address, view TOR statistics including TOR bandwidth, and more. Since Anonsurf uses TOR protocols for routing and full hop-by-hop encryption, it is comparatively slow relative to other VPNs, but provides better security. The AnonSurf GUI and command line interface menu are shown in Figs. 11 and 12, respectively.

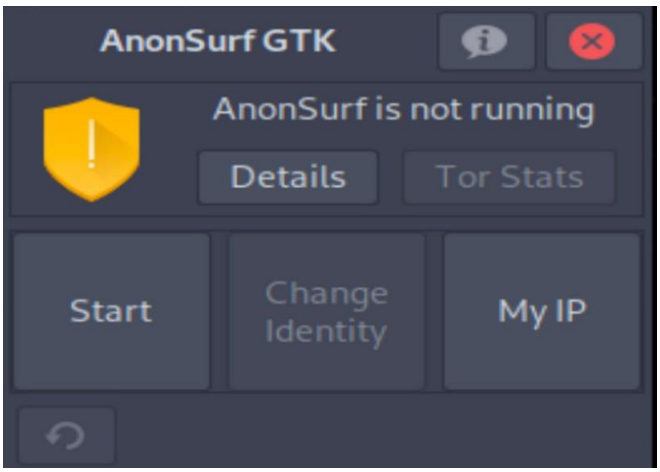

Fig. 11 AnonSurf Menu

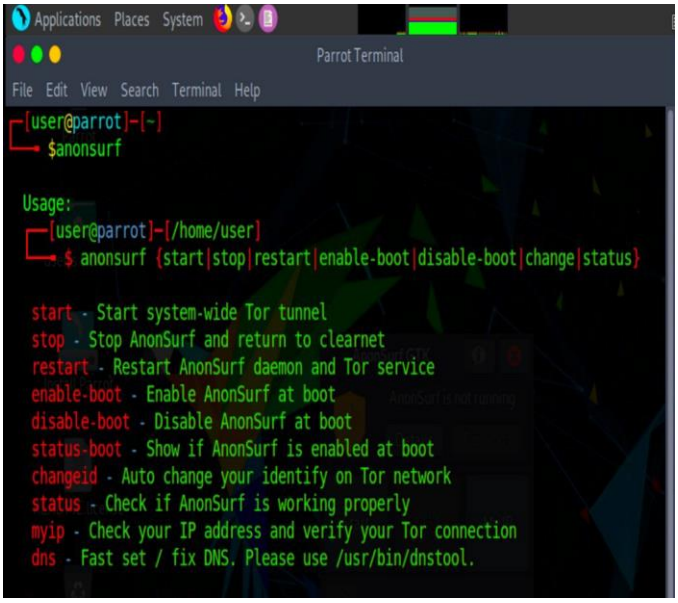

Fig. 12 AnanSurf command line interface

Once AnonSurf provides an IP address, you can optionally look up the actual location of this address using a variety of open source online geolocation services. In this way, you can request a different IP address in order to reach resources located in different parts of the world.

V. SUMMARY AND CONCLUSIONS

To address the growing number of cyberattacks, particularly against DNS servers, we configured a penetration test environment based on the open source Linux Distribution Parrot OS. We established that the widely used GUFW firewall is effective at blocking SYN scans in stealth mode with spoofed source IP addresses, FTP bounce scans, Nikto host vulnerability scans, and Metasploit basic scripting DoS attacks. We then determined that even in this environment, VPN connections are not necessarily secure, since there are opportunities for DNS leakage. We investigated approaches to mitigate this issue; while no anonymizing technique is completely foolproof, we obtained good results using the AnonSurf application bundled with Parrot OS to bypass DNS filtering and prevent information leakage. Future research in this area will include investigation of other Debian privacy tools supported by Parrot OS.

REFERENCES

[1] R. David, "The role DNS plays in network security", HelpNet Security, June 2021, https://www.helpnetsecurity.com/2021/06/07/dns-network-security/ (last accessed January 22, 2022)
[2] Parrot OS documentation, https://www.parrotsec.org/docs/welcome-to-parrot-doc.html (last accessed January 22, 2022)
[3] GUFW firewall documentation, http://gufw.org/ (last accessed January 22, 2022)
[4] ClamAV documentation, https://www.clamav.net/ (last accessed January 22, 2022)
[5] "Anti Virus - What Does This ClamAV Message Mean?" *Super User*, https://superuser.com/questions/956128/what-does-this-clamav-message-mean (last accessed January 22, 2022)
[6] Proton VPN documentation, https://protonvpn.com (last accessed January 22, 2022)
[7] Nikto documentation, https://cirt.net/Nikto2 (last accessed January 22, 2022)
[8] Metasploit documentation https://www.metasploit.com/ (last accessed January 22, 2022)
[9] "How Does DNS Filtering Work?" *Web Filtering*, 30 Aug. 2019, www.spamtitan.com/web-filtering/how-does-dns-filtering-work/(last accessed January 22, 2022) ; see also "4 Ways to Bypass Internet Filtering and Censorship." *Tech Times*, Tech Science Business Health Culture Features Buzz, 26 Aug. 2020, www.techtimes.com/articles/252068/20200826/4-ways-to-bypass-internet-filtering-and-censorship.htm. (last accessed January 22, 2022)
[10] DNS leak test application and documentation https://www.dnsleaktest.com/ (last accessed January 22, 2022)
[11] AnonSurf documentation, https://linuxhint.com/anonsurf/ (last accessed January 22, 2022)