CHAMPLAIN COLLEGE | LCDI Leahy Center for Digital Investigation

VPN/Proxy Chain Research

175 Lakeside Ave, Room 300A
Phone: (802)865-5744
Fax: (802)865-6446
http://www.lcdi.champlain.edu

12/18/2017

Disclaimer:

This document contains information based on research that has been gathered by employee(s) of The Senator Patrick Leahy Center for Digital Investigation (LCDI). The data contained in this project is submitted voluntarily and is unaudited. Every effort has been made by LCDI to assure the accuracy and reliability of the data contained in this report. However, LCDI nor any of our employees make no representation, warranty or guarantee in connection with this report and hereby expressly disclaims any liability or responsibility for loss or damage resulting from use of this data. Information in this report can be downloaded and redistributed by any person or persons. Any redistribution must maintain the LCDI logo and any references from this report must be properly annotated.

Contents

Introduction

With each passing day, privacy is becoming more of a concern for everyday people. There are many solutions for securing online privacy, but what if one's privacy solution of choice fails? The integrity of a network or computer device has never been more important. Maintaining privacy is just as important as maintaining security. With the growing number of companies experiencing breaches, increased online tracking for ad purposes, and continuing revelations about government surveillance programs it makes more sense for users to desire a degree of obfuscation in their online lives. Because users are trusting these systems, it's important to verify how those systems fail as well. This project explores the steps an individual can take to secure their private internet connection and create a failsafe system for online privacy. To do so, the VPN/Proxy Chain team used three Raspberry Pis to create a set of test proxies. Our research focused on providing an implementation that mimicked an end user using both a proxy network and a VPN to create a more secure internet connection. Our intentions are to create a network security system that has not only secure connections but does not allow the user to continue browsing in an unsafe manner if part of the system fails.

Background:

The LCDI has conducted prior research on the use of Raspberry Pis for various forensics and security applications. They have an inherent appeal due to their low price tag and portability. In the public sphere, there has been previous research conducted into porting various privacy solutions to the Raspberry Pi. Because they use a Linux based operating system, relevant background to this project also includes various Linux based privacy solutions. These solutions are not optimized for the Raspberry Pi, but remain relevant in the creation of new custom-made privacy solutions. The various desktop solutions provided a blueprint for how the Raspberry Pi based system should function, including best practice configurations and usage guidelines.

Purpose and Scope:

The purpose of this project is to see if creating a failsafe system for internet privacy is possible using a combined VPN and proxy implementation. It would be beneficial to use the two in combination because multiple layers of security make it less likely that a single point of failover will led to exposure. The scope of this project will include the research, implementation, and testing of the VPN/proxy solution we develop.

Research Questions:

1) What is the most user-friendly way to establish proxy and VPN services on the Windows operating system?
2) What is the most user-friendly way to establish proxy and VPN services on a Linux based operating system?
3) What type of proxy and VPN services can the Raspberry Pi support?
4) How do these methods fail gracefully when a proxy or VPN is removed from operation?

Terminology:

Virtual Private Network (VPN) - A a network connection that creates an encrypted connection to a VPN server, making it appear to whoever is watching that your traffic is coming from the VPN's IP address. All internet traffic from your computer uses the VPN encrypted tunnel. This prevents anyone from looking at your data on the trip between you and the VPN server.

Proxy - Creates a secure connection between your computer and the proxy server. Proxies were designed to encrypt only one application at a time. The proxy usually has to be configured for each application individually, and often passes the original IP address along. In a chain, proxies can provide a degree of anonymity.

Raspberry Pi - A low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a small, capable device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games. www.raspberrypi.org/help/what-%20is-a-raspberry-pi/

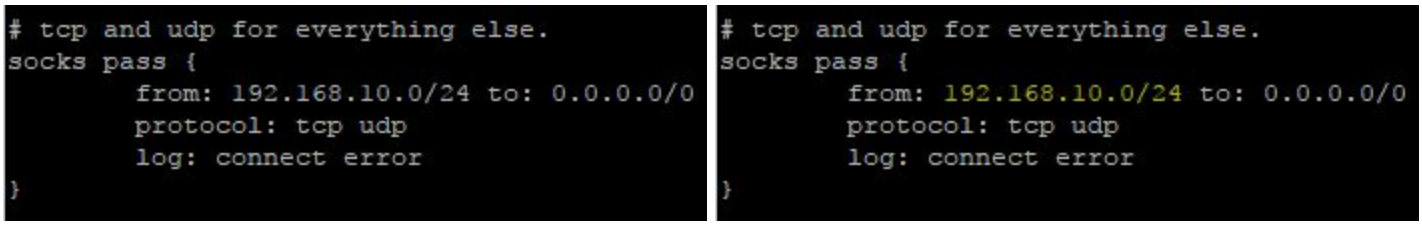
PiVPN - A set of shell scripts that serve to easily turn your Raspberry Pi into a VPN server using OpenVPN software.

Virtual Machine - A self-contained operating environment that behaves as if it is a separate computer, with no access to the host operating system. http://www.dictionary.com/browse/virtual-machine

Operating System - The collection of software that directs a computer's operations, controlling and scheduling the execution of other programs, and managing storage, input/output, and communication resources. http://www.dictionary.com/browse/operating-system?s=t

Methodology and Methods

For this project, we utilized three Raspberry Pi 3s as the privacy devices, an Ubuntu Linux client, a Windows 10 client, and an installation of Apache Server on a machine running Ubuntu Linux. All three Raspberry Pi 3's were flashed with Raspberry Debian Jessie as the operating system. Two of the Raspberry Pi 3s hosted Dante Proxy Servers. To do this, we ran the *sudo apt-get install dante-server* command on the two Pis that we used as Proxy Servers. From there, we adjusted the configuration file (*/etc/danted.conf*) by changing the port and IP addresses listed in the file to what we were using. These changes are noted in the screenshots below.



Setup screenshots from Dante Server

The third Raspberry Pi is set up with an OpenVPN distribution called PiVPN. Each of the Raspberry Pi's was connected to the network using an Ethernet cable. The initial setup was conducted using an HDMI connected monitor and combo wireless keyboard and mouse. Once the initial setup was complete, the Raspberry Pis were connected through an SSH connection over the network.

Due to network constraints, we required a local method of verifying the IP masking provided by the proxy servers. The Apache server allowed for immediate review of connection logs to document the masking process and troubleshoot issues in that respect. It also simplified our network configuration to keep all the traffic relatively local.

Each client was set up with a different proxy application. This was due to the constraints provided by using different operating systems. The Ubuntu VM was equipped with the program Proxy Chains. This program is designed to "proxify" applications that do not have built in proxy support. An example of this would be routing nmap scans through a proxy. On the Linux VM we also tested the built-in proxy support of Firefox and a popular Firefox extension called FoxyProxy. In Firefox, the proxy settings are located in the "Advanced" panel under the Options Menu. The "Network" tab is selected and the proxy settings can be set by selecting "Settings" for the connections section and editing the manual proxy settings. FoxyProxy can be installed on Firefox via the "Addons" menu.

On the Windows Client, the application Proxifier was used for proxy testing. Similar to Proxy Chains on Linux, this program allows for network applications that do not otherwise support proxies to be "proxified". Unlike Proxy Chains, Proxifier has a GUI. The Windows Client was also the machine we utilized to test the VPN. The clients tested were Viscosity and the OpenVPN client. The VPN suffered from more extensive networking issues than the proxy servers.

We began by using PiVPN to install OpenVPN on the Pi for us. We did this by running the command *curl -L* [*https://install.pivpn.io*](https://install.pivpn.io) | *bash* and going through the setup process with the settings that we wanted to use, such as IP address, default gateway, and OpenVPN port number. After PiVPN finished its installation process, we created .ovpn profiles by running the command *pivpn -a* and then filling in the information about the profile. After the profile was created on the Pi, we had to get it onto each of the client VMs. To do that we used the SCP command for the Linux client and WinSCP for the Windows client. We would then use those profiles to connect to the OpenVPN server.

Equipment Used

Table 1: Hardware

Device	OS Version
Raspberry Pi 3 (x3)	Raspberry Debian Jessie (Raspbian Jessie)
USB to Micro-USB Cable	N/A
Ethernet Cable (x3)	N/A

Table 2: Software

Software	Version	Comments
Raspbian Jessie	4.9.41	Manually Installed via Command Line
Microsoft Windows 10		Installed on 1 vSphere Virtual Machine
Linux Ubuntu	16.01	Installed on 2 vSphere Virtual Machines
PuTTy	0.70	Used to SSH into the Raspberry Pis
Proxy Chains	3.1	Used on Linux Client
OpenVPN	2.4.4	Used on Windows Client
WinSCP	5.11.2	Used on Windows Client to transfer files from Raspberry Pis
Proxifier Standard Edition	3.31	Used on Windows Client
Firefox	56.0	Used on Windows and Ubuntu Client
FoxyProxy	5.1.3	Used on the Ubuntu Client

Analysis

We analyzed the various settings for each application and how they failed. The general methodology for testing the resiliency of each setup involved configuring the client side proxy application in a Redundancy or equivalent mode. In a client side proxy application, this setting indicates that the application will pass network traffic through the proxies provided in a way that would utilize them all.

Using the Load Balancing or Redundancy settings on Proxifier (configured on the Windows machine), we tested how the Windows Proxy configuration responded to an ungraceful shutdown of the proxies. Using this configuration we discovered that if a proxy was removed from use, Proxifier would automatically divert the network traffic through the remaining viable traffic. This maintained the user's privacy by continuing to browse with a concealed IP address. When all proxies were taken down, Proxifier refused to allow network connection

and provided a “The connection has timed out” message. While this message is unclear for troubleshooting purposes, it does provide the promising result of preventing users from accessing the web when their pre-configured proxy settings are disabled. Proxifier was also tested using a “Simple Chain” setting, which required manually setting the order the proxies were to be accessed in. This means that the application would not provide any dynamic handling of the connections. When using a Simple Chain, if the first proxy goes down, the user is greeted with a “The connection has timed out” message regardless of the state of the other proxies.

The second proxy application tested was Proxy Chains configured on an Ubuntu client. Proxy Chains was configured to utilize the Dante servers as proxies and use the Apache test server as the destination to access. When just one Pi was taken down, the Proxy Chains application was still able to connect by routing traffic through the other proxy. When the connection to both proxies was lost, Proxy Chains was unable to connect out to the Apache server. This showed was that as long as there is a functioning proxy, Proxy Chains will be able to connect, but without any up proxies, our setup will fail to connect.

The Proxy Chains program we used on the Ubuntu Linux VM was very precise during set up.. If all of the proxies listed in the chain were not in the correct order with their respective servers on, it would fail to connect to Firefox or the individual proxies within the chain.

We also conducted tests with the Firefox browser and the FoxyProxy extension using our Raspberry Pis as the proxy servers. In Firefox standard proxy settings, the shutdown of the proxy was handled gracefully and the user was informed they were no longer connected to the internet. The same response was given by the FoxyProxy extension. At the time, the standard Firefox proxy function only allowed for one manually configured proxy with no redundancy. FoxyProxy was only tested using a single proxy as well, although it allowed for a redundancy setting.

Ultimately, when using the VPN we ran into a number of connection issues. It would appear the connection was made, but the IP address of the client would show up in the host’s logs. This obviously does not mask the user’s identity to any degree and we ran into a number of issues discerning as to why the connection was not being correctly initiated. Neither client side application (OpenVPN or viscosity) displayed any obvious sign to the user that the connection was not made correctly. After an extensive troubleshooting process, we believe that the issues resulted from the local network environment.

Results

This project verified how both Proxy Chains and Proxifier would handle ungraceful shutdown of a configured proxy that was currently in use. When either application was configured in a load balancing or equivalent manner, the removal of a single proxy would not negatively impact the user’s experience. The proxy application would simply route traffic through the remaining proxy and continue browsing. If all configured proxies were taken down, the proxy application would no longer allow the user to navigate to new pages.

In a configuration with a static chain of proxies in which the user specified what order the proxies were to be accessed in, the results were different. Any proxy removed in this setup would result in the user being unable to send network traffic.

This project also verified that Firefox's built in proxy handling application and the FoxyProxy extension would also handle the removal of proxies in a graceful manner that protected the end user.

Conclusion

This project suffered based on the network constraints. We were only able to reach solid conclusions on our chained proxy setup and find results on how a few types of client side programs dealt with ungracefully losing a proxy within the chain.

We can conclude that proxies should be kept to their original purpose of only concealing the activity of a singular application on the host. The best way to "proxify" all of the network traffic coming from the computer without extensively configuring each program is to simply use a VPN. We concluded that the applications tested for this purpose, Proxy Chains and Proxifier, both provided protection to the end user if the proxies went down during usage.

We can conclude that trusting mature and developed applications, such as those included in the web browser, is a strong way for users to be sure that if their proxies are taken offline, they won't continue browsing unknowingly. Using a built in browser extension had a more intuitive setup process that configuring a separate application to manage proxies for web usage. For single applications with no built in proxy support, third party applications can provide temporary masking that would also terminate connections if the proxies went down.

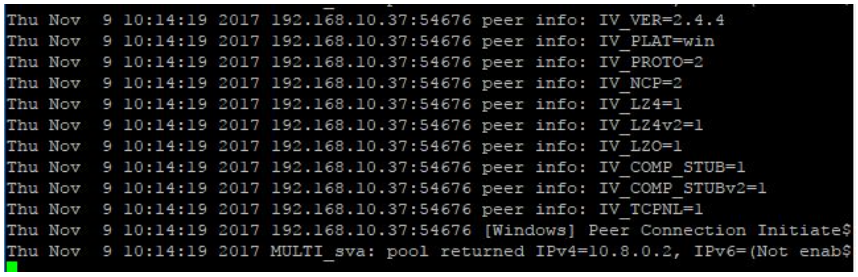
Further Work

Recommended next steps for this project would include deploying it in a sub-network that the research team has full control over, in order to more appropriately mimic an end user environment. There is also the option of configuring the proxies in a cloud environment, such as AWS or Digital Ocean, to mimic how proxies might be used on a more international scale.

It would also be interesting to combine different proxy server applications in a single chain, as currently all of our proxies are Dante Proxy servers. It would also be useful to investigate a fuller range of client side proxy programs since that is where there is most likely going to be an issue handling an ungraceful shutdown. Setting up a full suite of potential applications to combine with different server types would be a whole new set of research itself.

The VPN is an area that requires further research in order to reach a solid conclusion on how it would react when the VPN was taken down. In the future, it would be recommended to start off hosting the VPN in a cloud provider such as Digital Ocean or AWS to circumvent the local network issues. It would be worth investigating other free, build-it-yourself VPN options and how easy they are to deploy securely within a network. Once a

functional VPN that masks the user's IP address is established, then the VPN infrastructures can be investigated for flaws.



VPN logs

Once each aspect has been subjected to further research, it would be possible to get to the original objective of this project: find a solution that combines proxy and VPN servers for the maximum amount of user privacy. In this situation, the network activity for the entire computer would be sent through a VPN and individual applications be configured to use specific chains of proxies. Developing a way to asses which hop took precedence would be an interesting proposal. This would most likely require having access to remote, public web facing servers to use for the proxy and VPN providers. It is highly unlikely that in a user situation all of the devices will be on the local network, as that would defeat the point of using such protections in the first place.

References

(2017). *What is a Raspberry Pi?*. Retrieved from www.raspberrypi.org/help/what-%20is-a-raspberry-pi/

(2017). *How about a SOCKS proxy?*. Retrieved from https://www.privacypi.org/how-about-a-socks-proxy/

dayz(2017, Jan 22). *How to turn your raspberry pi into a home vpn server using pivpn.* Retrieved from http://kamilslab.com/2017/01/22/how-to-turn-your-raspberry-pi-into-a-home-vpn-server-using-pivpn/

redfast00(2017, Jun 29). *About PiVPN.* Retrieved from https://github.com/pivpn/pivpn

Random House, Inc. (2017). Virtual Machine. Retrieved November 30, 2017, from http://www.dictionary.com/browse/virtual-machine

Random House, Inc. (2017). Operating System. Retrieved November 30, 2017, from http://www.dictionary.com/browse/operating-system?s=t

Etcher by resin.io. (2017). Retrieved November 30, 2017, from https://etcher.io/

(2016, Dec 27). *Linux Proxy Server Settings - Set Proxy for Command Line.* Retrieved from https://www.shellhacks.com/linux-proxy-server-settings-set-proxy-command-line/

Burke, S(2006, Dec 1). *Dante Socks Server.* Retrieved from http://wiki.kartbuilding.net/index.php/Dante_Socks_Server