Application of Stream Splitting Moving Target Network Defenses

Joshua Klosterman
Beacom College of Cyber and Computer Sciences
Dakota State University
Madison, SD, USA
Joshua.Klosterman@dsu.edu

Jacob L. Williams
Beacom College of Cyber and Computer Sciences
Dakota State University
Madison, SD, USA
Jacob.Williams@dsu.edu

Michael C. Shlanta
Beacom College of Cyber and Computer Sciences
Dakota State University
Madison, SD, USA
Michael.Shlanta@dsu.edu

Daniel W. Burwitz
Beacom College of Cyber and Computer Sciences
Dakota State University
Madison, SD, USA
Daniel.Burwitz@trojans.dsu.edu

***Abstract*—Our world is ever changing, and cyber adversaries are continually finding new and innovative ways to bypass existing defenses. This paper investigates the application of a Moving Target Network Defense and demonstrates a capability to impair an adversary attempting to execute a successful cyber-attack. This paper also reviews previous research into Moving Target Defenses. While moving target defenses do pose some difficulties in implementation, the benefits could be significant.**

Keywords—Moving target defenses, Multipath TCP, cyber-attack

I. INTRODUCTION

A. Problem

In the United States, modern internet protocols, such as TCP, typically send their communication payloads over a single route over the Internet. In turn, cyber attackers can intercept these communications by "sniffing" and capturing packets on one of these network routes, compromising user privacy as well as data integrity. Ideally, modern cyber communication protocols should not send their entire communication payloads over a singular network route to prevent malicious users from being able to steal an entire communication payload.

B. Purpose

To deter attackers from possibly stealing user data over the Internet and to protect user privacy, the team's research considers feasible, programmatic implementations of sending TCP streams over multiple network paths using Multipath TCP. This way, attackers cannot obtain an entire communication payload even if they are able to capture packets during a segment of a network path.

C. Motivation

When TCP/IP was originally conceived, it was intended to facilitate network communications between academic institutions; as such security was not a concern when these protocols were built. However, as the world becomes ever more reliant on digital communications and the Internet as a primary means of communication and data storage, information security has become an important area of research. Unfortunately, as modern cyber infrastructures still use these legacy protocols for communication and cyber attackers find more ingenious ways to steal information, researchers must find solutions to ensure that users' data is protected from unwanted eyes.

To stay ahead of cyber attackers, researchers in the past have delved into research on Moving Target Defense. For example, Random Host Mutation is one type of defense that has been researched as early as 2012 [1]. However, for implementations of Moving Target Defense, splitting TCP streams is a new area of research, with little work to build the team's research approach. Therefore, the team's research attempts to create a starting point for future researchers who wish to research this area and build a workable implementation of splitting TCP streams over multiple network routes. Hopefully, the team's research can kickstart future research into using Multipath TCP as a Moving Target Defense methodology that can be used in the public Internet to protect user data from cyber attackers.

II. EXECUTIVE SUMMARY

In a traditional cyber-attack, an attacker takes steps such as reconnaissance and network mapping to gather information about their target to discover vulnerabilities and determine their method of attack. During the critical reconnaissance phase of their attack, an attacker relies on the static nature of a cyber infrastructure to gain valuable insight into their target, such as operating systems, IP addresses, and network configurations. A Moving Target Defense (MTD), attempts to continuously rotate and randomize the information departing the network, increasing the difficulty of launching a successful attack or stealing information. While several different types of MTDs have been developed, including Random Host Mutation, Address Space Layout Randomization, and others, this project focused on developing a new type of MTD that utilizes multiple TCP streams to decrease the risk of information leakage during a data transmission. Also known as Split-stream TCP, the team aimed to send TCP streams over the internet using different routes and paths, making it harder for an attacker to intercept any useful information.

III. LITERATURE REVIEW

One work that was used to establish a knowledge base was 'Computer Network Deception as a Moving Target Defense'. This paper mostly provided insight into how MTDs obscure information for attackers, such as IP and port overlays[2]. It also gave examples of deception at the

Work supported by the National Science Foundation under Grant No. 1344369

application layer, which is much more applicable to the current scenario, since that's where the team's program operates. Their proposed MTD involved things such as host attributes, characteristics, and files[2]. They also discussed the different negative impacts on attackers that the deception framework of their network hoped to achieve, such as: obscuring the real target(s), devaluing information, causing the adversary to waste time and resources, forcing the adversary to reveal advanced capabilities, exposing adversary intent, increasing difficulty of attack planning, limiting the scope of the attack, and limiting the duration of a successful attack[2]. These are important objectives in denying access to information to any attacker. These are rather clear-cut and are certainly objectives this application does/will address.

Another paper reviewed by the team was 'Characterizing Network-Based Moving Target Defenses'. It discussed four Network-Based Moving Target Defenses (NMTDs), common properties they should have used, and which ones they utilized.

There were three key properties which are vital to the operation of a successful network moving target defense system: the **Moving property** (the system must be *unpredictable* in its movement, must have a *vast destination space*, and must have *periodicity*, meaning it moves with regularity), the **Access Control property** (guarantee *uniqueness*, or that each client is individually authorized and cannot be shared with any other client, *availability*, which states that if a client is authorized, it can successfully reach the target when desired, and *revocability*, which allows the NMTD's mapping system to terminate or expire previous authorization without collateral damage), and the **Distinguishability property**, (the system can separate trustworthy clients from untrustworthy ones)[3].

Four proposed NMTD's were evaluated by the authors: an approach utilizing a DNS server and a NAT device, one using OpenFlow Mutation for address randomization, a Moving Target IPv6 Defense (MT6D) system, and Simulation-based MTD [2]. Here we'll give a brief description of each approach, and which properties that it does and does not meet.

The server IP Address is provided to a client with a short time-to-live value in the DNS record, and the NAT device uses this response to create a window on that IP address for that client to connect. If that client connects in the allotted time, the NAT device creates a mapping for the client to reach its target. By using a pseudo-random number generator and IPv6 addresses, this approach achieves the three sub-properties required for the Moving property. Due to DNS caching, two clients could potentially share a return address, but it would be difficult for an adversary to do so. This would require an attacker to use a client from the same subnet, guess the right IP address from the destination range, and issue a connection in the same Time To Live window, which is a short amount of time in this implementation. Then, since NAT devices already track flows, this approach does not hinder availability, and to revoke network access, any NAT mapping between the target and client can be removed, which will automatically redirect the client to a sink. Therefore, the Access Control property is met. These first two properties are met by the DNS approach[3].

The drawback with this DNS approach comes in meeting the Distinguishability property. The DNS capability cannot distinguish between automated attackers that immediately request and use DNS, and legitimate clients. This creates a weakness in this implementation[3].

The OpenFlow mutation approach uses IP address mutation in a LAN using the OpenFlow protocol. This protocol allows network switches to be configured as high-speed network caches. If the switch does not know how to forward a packet, it asks the OpenFlow controller for instructions[1]. This is called the *elevation* mechanism and is used to alter DNS records and change packet addresses mid-flight. Whenever a client performs a DNS lookup, it is assigned a virtual IP address from a pool of IP addresses, which are selected at random or in a weighted random fashion using only information known to defenders, thereby meeting the unpredictability and periodicity requirements of the Moving property. Vastness is also achieved in both IPv4 and -v6, since even if an adversary enumerates possible IPv4 combinations, the OpenFlow controller can detect this and block the host, and therefore achieves the Moving property. When the client attempts to access the target, the controller can distinguish requests from clients without an intermediary DNS resolver acting as a proxy, fulfilling the uniqueness requirement of the Access Control property. The revocability requirement is also achieved since the controller can replace flow translations with drop rules to terminate connections between machines. The availability requirement is met if the controller and switches can handle traffic elevation and forwarding. These determine scaling in these OpenFlow networks, and some constraints were not directly expressed, so it is unknown if the final requirement is fully met, so therefore the Access Control property is not fully met. Virtual IP addresses are used to forward packets until the destination switch is reached, where the packet is translated to the destination host's real IP address. In this implementation, the switches are effectively NAT devices and the OpenFlow controller serves as a mapping system. Like the DNS implementation, the OpenFlow mutation implementation can only distinguish between clients that use DNS and those that are performing scans, so the Distinguishability property is not fully met. This is really only suited to a LAN since the controller must have knowledge of the source and destination switches[3].

In the MT6D implementation, the client and host share a symmetric key out-of-band and use them to generate IPv6 addresses for those hosts to use. Using this shared key, a hash is created, 64 bits are extracted from the hash, and used along with a value pulled from the host's MAC address and a timestamp, their IPv6 addresses are created, with the bits from the hash forming the lower 64 bits of the address. This meets the unpredictability requirement, the rapid transition among addresses meets the periodicity requirement and using the IPv6 address space meets the vastness requirement, thereby meeting the needs for the Moving property. The authors of the original proposal never specify if the shared key is unique. If it is, it meets the uniqueness and revocability requirements, otherwise (if it is not unique) it fails to meet this need. Also, they do not show that the availability requirement is met, most notably the impact address rotation could have on network infrastructures near the host. However, the authors of the analysis paper do give ideas on how to make it completely meet the requirements for the Access Control property. This includes the need for MT6D to have a unique symmetric key at both hosts, so this approach can discriminate between unauthorized users who lack said keys[3].

In the Simulation-based MTD, the clients and targets were modified with MTD software that directed host movement. Mapping services were on the individual hosts and targets. The process of connecting from client to target is discussed by either proxying (like in MT6D) or using a special API. By making movement chaotic, this meets the unpredictability requirement. However, constraints needed to meet the other two requirements for the Moving property were not explicitly discussed. The requirements for the Access Control property were not explicitly mentioned either but could be partly met by avoiding duplicated roles across clients and targets. This system can detect differences between authorized systems with pre-configured MTD software and systems that do not possess these configurations[3].

Three of the four evaluated NMTDs did not address at least one key property that was mentioned above. For the NMTD that does *theoretically* meet all properties, the original authors for the NMTD did not specifically mention how the NMTD would directly address each property. The authors who analyzed the implementation suggest how it *could* meet the unmentioned properties. While this implementation has the potential to meet the required conditions, it does not explicitly meet them. This leaves a glaring weakness in one or more parts of these systems and makes it impossible to determine if these implementations are viable, and the biggest takeaway from this was that it is difficult to create an NMTD that possesses each sub-property and property that a good NMTD should have.

Despite finding many more research articles than discussed above, there was very little the team could use in the way of borrowing work for additional research. All the MTD approaches that the team found take place at the Data Link, Network, and Transport layers of the OSI model. This includes anything from randomization of MAC and IP addresses, port overlays, or Multipath TCP. In this project, the group was asked to design an application-level implementation of Multipath TCP. That idea already goes beyond the scope of previous MTD implementations. While the application does interact with the Transport layer, it is done through the application layer. This has allowed us to make changes in the application layer and let all other layers handle themselves normally. This was done due to time constraints making it difficult to use a lower-level language to better work at these layers. While these articles and pieces of research greatly improved general understanding of MTDs, the team was largely left to blaze their own trail into application-level MTD.

IV. FINDINGS

A. Findings Overview

Our current implementation has been run and tested in a live environment. It is most effective if used with multiple bouncer nodes. This causes the information to take longer to get to the receiver, but utilizes a much more diverse path. It is still possible for the paths to converge onto one but will most likely happen directly before the receiver, so using different mediums to send the communication will increase path diversity. See Fig. 1 and Fig. 2 to see examples of path diversity during the project. The current version will split the file into at least two parts and send the different parts out to different starting bouncers. The receive node will be in a listening state until it has been notified that the file has been fully sent. After this, the file sent will be reassembled in the correct order. See Fig. 3 for a simplified model of the implementation network.

V. CONCLUSIONS AND RECOMMENDATIONS

A. Conclusions

The team found that employing split TCP streams as a Moving Target Defense methodology is a programmatically viable way to protect user data over the public Internet. The team was able to get a fully functional proof-of-concept built and tested within the project timeline. The team verified multiple routes through Traceroute executions that sending data over different networks provides diverse paths to their destinations (see Fig. 1 & Fig. 2). Therefore, it would be difficult for a cyber attacker to obtain a full communication payload unless they had packet sniffing programs on multiple routes, which would be only accessible to cyber attackers with great resources.

B. Recommendations

First, the team believes that not sending all TCP communication traffic through the Internet will avoid traffic bottlenecks at single routes. Instead of sending TCP traffic through different networks, the traffic could be sent over different communication mediums, such as satellite and cellular networks. Due to communication traffic not being sent over the same medium, the possibility of network traffic bottlenecking at a single router is eliminated and the difficulty for an attacker to intercept the entire communication payload increases.

Finally, the last recommendation is for research that may completely redesign and implement split stream TCP as a Moving Target Defense methodology. When writing the program, the group recommends a lower level language than Python. Due to the high-level nature of the language, the team wasn't able to gain a fine-grained of control over the TCP/IP protocols to work exactly as the team hoped. Therefore, creating the program in a lower level language such as C would give a finer level of control over the networking, though the programming task would require experienced programmers. Second, the team would suggest implementing encryption on each of the file segments as an extra layer of security, in cases where an attacker manages to get ahold of the entire communication payload. Additionally, the team would add Traceroute-like functions to the program to check for path diversity, and to warn users if there exists a route where the network paths converge. This way, users are aware if their TCP streams converge on a single route.

DISCLOSURES

Joshua Klosterman is currently an employee of The MITRE Corporation. The author's affiliation with The MITRE Corporation is provided for identification purposes only and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions, or viewpoints expressed by the author. (Approved for Public Release; Distribution Unlimited Case 18-3380)

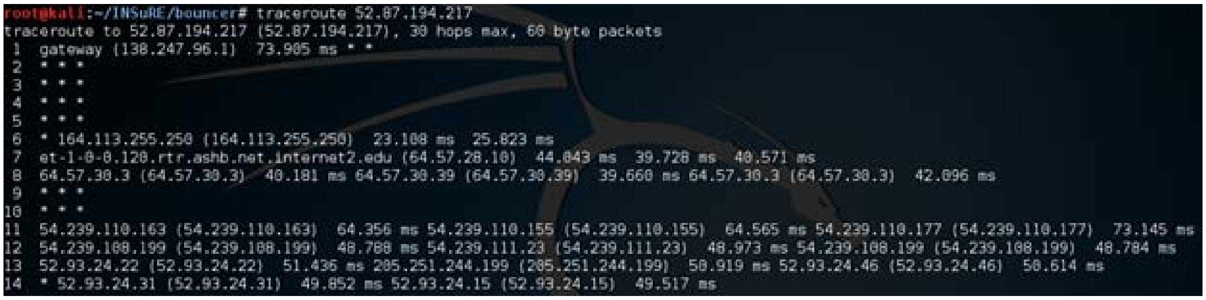
REFERENCES

[1] J. H. Jafarian, E. Al-shaer, Q. Duan, and C. C. N. Network, "OpenFlow Random Host Mutation : Transparent Moving Target Defense using Software Defined Networking," pp. 127–132.

[2] V. E. Urias, W. M. S. Stout, and C. Loverro, "Computer network deception as a Moving Target Defense," Proc. - Int. Carnahan Conf. Secur. Technol., vol. 2015–January, 2016.

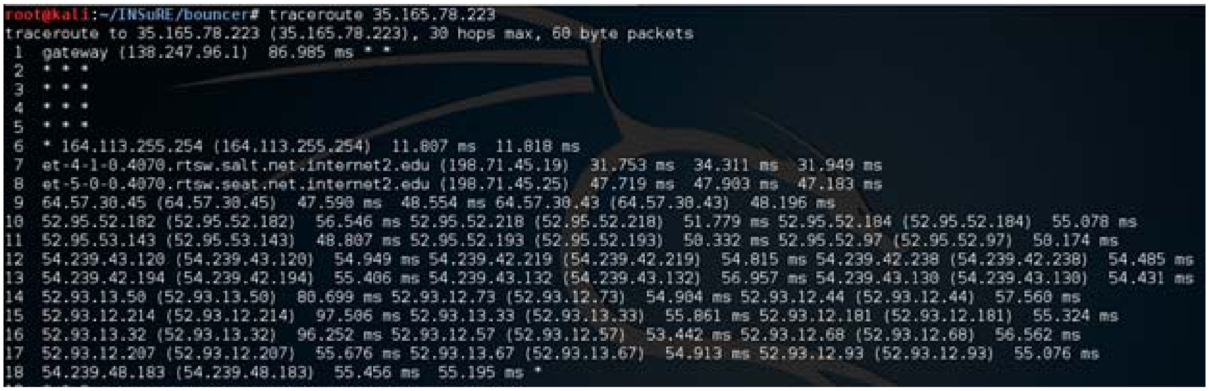
[3] M. Green, D. C. Macfarland, D. R. Smestad, and C. a Shue, "Characterizing Network-Based Moving Target Defenses," Proc. 2nd ACM Work. Mov. Target Def., pp. 31–35, 2015.

FIGURES



```
root@kali:~/INSuRE/bouncer# traceroute 52.87.194.217
traceroute to 52.87.194.217 (52.87.194.217), 30 hops max, 60 byte packets
 1  gateway (138.247.96.1)  73.905 ms * *
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * 164.113.255.250 (164.113.255.250)  23.108 ms  25.823 ms
 7  et-1-0-0.120.rtr.ashb.net.internet2.edu (64.57.28.10)  44.043 ms  39.728 ms  40.571 ms
 8  64.57.30.3 (64.57.30.3)  40.181 ms 64.57.30.39 (64.57.30.39)  39.660 ms 64.57.30.3 (64.57.30.3)  42.096 ms
 9  * * *
10  * * *
11  54.239.110.163 (54.239.110.163)  64.356 ms 54.239.110.155 (54.239.110.155)  64.565 ms 54.239.110.177 (54.239.110.177)  73.145 ms
12  54.239.108.199 (54.239.108.199)  48.788 ms 54.239.111.23 (54.239.111.23)  48.973 ms 54.239.108.199 (54.239.108.199)  48.784 ms
13  52.93.24.22 (52.93.24.22)  51.436 ms 205.251.244.199 (205.251.244.199)  50.919 ms 52.93.24.46 (52.93.24.46)  50.614 ms
14  * 52.93.24.31 (52.93.24.31)  49.852 ms 52.93.24.15 (52.93.24.15)  49.517 ms
```

Fig. 1. Traceroute 1



```
root@kali:~/INSuRE/bouncer# traceroute 35.165.78.223
traceroute to 35.165.78.223 (35.165.78.223), 30 hops max, 60 byte packets
 1  gateway (138.247.96.1)  86.985 ms * *
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * 164.113.255.254 (164.113.255.254)  11.807 ms  11.818 ms
 7  et-4-1-0.4070.rtsw.salt.net.internet2.edu (198.71.45.19)  31.753 ms  34.311 ms  31.949 ms
 8  et-5-0-0.4070.rtsw.seat.net.internet2.edu (198.71.45.25)  47.719 ms  47.903 ms  47.183 ms
 9  64.57.30.45 (64.57.30.45)  47.590 ms  48.554 ms 64.57.30.43 (64.57.30.43)  48.196 ms
10  52.95.52.182 (52.95.52.182)  56.546 ms 52.95.52.218 (52.95.52.218)  51.779 ms 52.95.52.184 (52.95.52.184)  55.078 ms
11  52.95.53.143 (52.95.53.143)  48.807 ms 52.95.52.193 (52.95.52.193)  50.332 ms 52.95.52.97 (52.95.52.97)  50.174 ms
12  54.239.43.120 (54.239.43.120)  54.949 ms 54.239.42.219 (54.239.42.219)  54.815 ms 54.239.42.238 (54.239.42.238)  54.485 ms
13  54.239.42.194 (54.239.42.194)  55.406 ms 54.239.43.132 (54.239.43.132)  56.957 ms 54.239.43.130 (54.239.43.130)  54.431 ms
14  52.93.13.50 (52.93.13.50)  80.699 ms 52.93.12.73 (52.93.12.73)  54.904 ms 52.93.12.44 (52.93.12.44)  57.560 ms
15  52.93.12.214 (52.93.12.214)  97.506 ms 52.93.13.33 (52.93.13.33)  55.861 ms 52.93.12.181 (52.93.12.181)  55.324 ms
16  52.93.13.32 (52.93.13.32)  96.252 ms 52.93.12.57 (52.93.12.57)  53.442 ms 52.93.12.68 (52.93.12.68)  56.562 ms
17  52.93.12.207 (52.93.12.207)  55.676 ms 52.93.13.67 (52.93.13.67)  54.913 ms 52.93.12.93 (52.93.12.93)  55.076 ms
18  54.239.48.183 (54.239.48.183)  55.456 ms  55.195 ms *
```

Fig. 2. Traceroute 2

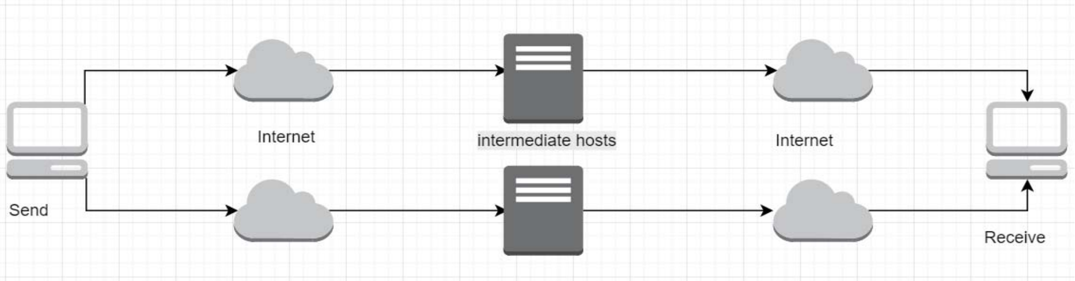


Fig. 3. Simplified network implementation model