Moving Target Defense Router: MaTaDoR

Berkan Ufuk[1,2][a] and Mehmet Tahir Sandikkaya[1][b]

[1]*Computer Engineering Department, Istanbul Technical University, TR34469, Turkey*

[2]*Aselsan Inc., Ankara, Turkey*

Keywords: Moving Target Defense, TCP-authentication Option, Transparent Routing, Message Authentication, DoS, Honeynets, Honeypot.

Abstract: The continuous increase in network attacks and the complexity of the available offensive technologies enforces novel defensive mechanisms. Moving Target Defense (MTD) is a recent family of approaches for network defense. This study proposes MaTaDoR, which utilizes message authentication akin to TCP Authentication Option (TCP-AO) in a MTD setting to mitigate a wide range of attacks, including Denial of Service (DoS). The purpose of MaTaDoR is averting unauthenticated packets from reaching protected assets. When many other MTD approaches aim to delay adversaries, MaTaDoR strictly protects networked assets from unauthenticated access. MaTaDoR is transparent, stateless and scalable. The efficiency of this combination is demonstrated by the results of a simulation. The proposed approach is capable of blocking every DoS packet with an insignificant trade-off increase in end-to-end delay.

1 INTRODUCTION

Although there are numerous network defense mechanisms, the number of attacks is increasing. New defense mechanisms evolve hand in hand with new attack strategies. Generally, the motivation behind a defense mechanism is to make an attack relatively costly for an adversary. Moving Target Defense (MTD) is one of the mechanisms in order to protect a networked service from attacks. MTD aims to confuse adversary by manipulating network attributes in order to render attacks infeasible. This study contributes by proposing eager message authentication for networks that utilize MTD. The proposed solution is essentially similar to chaffing and winnowing (Rivest et al., 1998). Host-independent message authentication chaffs stray TCP connections. TCP Authentication Option (Touch et al., 2010) has common usage in networks linked with Border Gateway Protocol (BGP). The hypothesis is that considerable amount of unnecessary requests could be prevented prior to establishing connections if MTD with an authentication header (abbreviated as MaTaDoR – Moving Target Defense Router) is utilized. Selection and elimination of packets are possible in the beginning of the TCP-handshake step by inspecting how the SYN packets are constructed. Therefore, only genuinely message-authenticated SYN packets would be apt to be processed and other packets would be directed to a decoy network. The proposed mechanism resides in a transparent box to route the traffic to either the genuine or the decoy network that have equivalent structure. The deployment of the proposed mechanism is shown in Figure 1.

The mechanism is effective against all kinds of unauthorized access attempts. DoS, where such behavior is observed frequently, forms a valid demonstration. Therefore, the effectiveness of the hypothesis is shown by simulating the mechanism with regenerated real-world network traffic which is known to include DoS attacks.

MaTaDoR is a router, rather than a gateway. It does not keep the state of the connections that are passing through so that how the unwanted connections could be relayed or dropped before handshakes are completed. Therefore, MaTaDoR itself is resistant against overflows. Since the transparent entry point in the proposed mechanism acts as a router, it does not form a single point of failure. Scalability is achieved by adding or removing MaTaDoR boxes. Central key management is not necessary as MaTaDoR is stateless.

The proposed approach is compared with ongoing research in Section 2. Section 3 sketches a typical use case of the proposed solution. Re-generation of the publicly available real-world traffic for the conducted

[a] https://orcid.org/0000-0002-1416-2866
[b] https://orcid.org/0000-0002-9756-603X

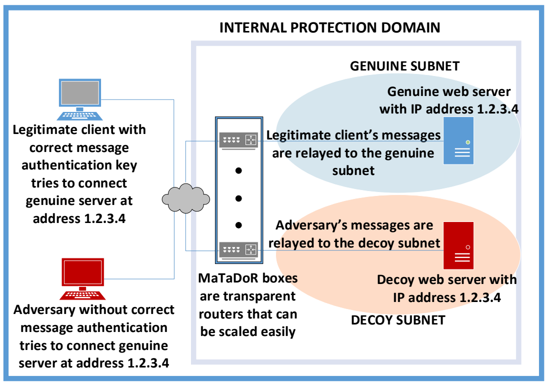

INTERNAL PROTECTION DOMAIN
GENUINE SUBNET
Genuine web server with IP address 1.2.3.4
Legitimate client with correct message authentication key tries to connect genuine server at address 1.2.3.4
Legitimate client's messages are relayed to the genuine subnet
Adversary's messages are relayed to the decoy subnet
MaTaDoR boxes are transparent routers that can be scaled easily
Decoy web server with IP address 1.2.3.4
DECOY SUBNET
Adversary without correct message authentication tries to connect genuine server at address 1.2.3.4

Figure 1: MaTaDoR is able to distinguish the legitimate packets from malicious ones even though their observable behavior is the same for an external observer.

simulation and reproducible experimental steps are explained in Section 4. The results of the conducted simulations for the proposed cost-effective mechanism are tabulated in Section 5. Section 6 concludes the paper with a summarizing discussion.

2 RELATED WORK

Hong, et al. classifies MTD techniques into three categories: Shuffle, Redundancy and Diversity (Hong and Kim, 2015). *Shuffling* moves or re-arranges the resources to be protected. *Redundancy* replicates the resources to enhance availability. *Diversity* places different variants of a resource with the same functionality. According to this taxonomy, proposed mechanism is a diversity mechanism as the adversary is diverted to a non-functional decoy.

Chaffing and winnowing is a technique for sharing confidential information in an untrusted medium (Rivest et al., 1998). Chaffing and winnowing was not designed as a MTD approach. It is designed to provide confidentiality through message authentication instead of encryption. The proposed mechanism was inspired by chaffing and winnowing as message authentication is used as an effective filter to distinguish legitimate or malicious traffic.

Bogosyan et al. uses MTD to do defend against spoofing and replay attacks to Controller Area Network (CAN) bus during electric vehicle charging process (Bogosyan et al., 2020). This study also establishes redundancy MTD by multiplying resources.

Jenkins et al. design a MTD algorithm for real time networks in space systems (Jenkins et al., 2021). This study makes use of authentication bytes and distributes correct shuffling keys to authenticated clients. This study designs a shuffling MTD.

Park, et al. propose Ghost MTD, where pre-shared one time bit sequences are distributed and non-conforming sequences are directed to some decoy module (Park et al., 2020). Park, et al.'s approach resembles the proposed mechanism; however, distribution and management of one time bit sequences for each host seems impractical.

Kampanakis et al. implement Moving Target Defense mechanism in a Software-Defined Network (SDN) environment, then evaluate the performance of the system with Cisco's One Platform Kit (Kampanakis et al., 2014). Even though the authors effectively use SDN's flexibility, it is difficult to adopt this approach to traditional networks.

Openflow Random Host Mutation (OF-RHM) is also based on SDN (Jafarian et al., 2012). Virtual addresses of network resources are continuously modified by the SDN controller via Openflow protocol. This is an example of a shuffling mechanism. The IP configuration of the network domain is constantly changing. On the other hand, the proposed mechanism has two different domains which have the same configuration from an outsider's point of view.

Network-based MTD (NMTD) constantly changes the network configuration such as IP addresses and ports that are used by the protected applications (MacFarland and Shue, 2015). This setup requires a centralized configuration to follow. However, this might introduce a single point of failure. On the other hand, MaTaDoR could be multiplexed without burden.

Luo, et al. proposed Random Port and Address Hopping (RPAH) scheme that predefines the active ports (Luo et al., 2017). A gateway handles the generation and management of virtual IP addresses. Luo et al.'s NAT implementation is similar to the proposed mechanism. However, the transparent entry point in the proposed mechanism acts as a router and does not form a single point of failure.

3 PROPOSED MECHANISM: MaTaDoR

The users of a network expect availability. A network is constantly subject to attacks that jeopardize availability. Therefore, networks should distinguish adversaries from legitimate users. In the proposed mechanism adversarial packets are routed to a decoy network right after they are distinguished. Thus, even if an attack is going on, legitimate users are served. Perceiving a decoy but equivalent service is confusing for adversary.

Considering these, MaTaDoR is designed as a transparent router. The decoy network in this study is presented as a placeholder and is not detailed. Else, it is possible to use this network to redirect adversaries or observe their actions further on a honeynet.

Slowloris and similar DoS attacks harm the availability of a web server even before connections are established. Since the attack traffic will not include the correct message authentication keys, MaTaDoR forwards this traffic into a decoy server. Even if the decoy server rendered unavailable, the legitimate users of the genuine server are not affected. However, from the adversary's point of view, disrupting the service of the website make them think their attack is successful at first sight. Such additional confusion is another layer of ambiguity for the adversary.

The relaying nature of MaTaDoR is transparent as seen in Figure 1. The sole functionality of MaTaDoR is to verify the message authentication header and determine the future flow of the traffic. A realistic scenario is to place MaTaDoR in between two organizations that share non-public network services without a dedicated VPN (Virtual Private Network) or a leased-line. Message authentication key is assumed to be set up during initialization. Further key management issues are deliberately keep out of the scope of the discussion. A second, decoy server is also placed in Figure 1 to handle attacker attention whenever MaTaDoR fails to authenticate the message. Moreover, the decoy server could be replaced with an honeynet that logs attacker behavior or non-verified segments could simply be dropped.

The network topology presented in Figure 1 represents the aforementioned scenario with two clients, two web servers and MaTaDoR. One of the clients (marked red in Figure 1) represent the malicious IP space where DoS attack is originating from. The packets from the malicious IP space have missing or incorrect MAC. The other client (marked blue in Figure 1) represent the benign users who want to use the web server. The genuine web server represents the protected resource. Finally, the decoy web server is equivalent to the genuine resource, but consists of mechanisms to distract adversarial efforts. As in every other MTD solution, the adversary is hopefully tricked to think that no defense mechanism exists to protect the asset.

The flow is shown in Figure 2a when a segment is authenticated. MaTaDoR verifies the authentication header, then directs the traffic to the web server if the MAC verification is successful. The genuine web server gets the segments and continues by sending an acknowledge message to establish TCP handshake with the authenticated client. The genuine web server can directly see the initiator IP address of the client, as MaTaDoR is transparent. Thus, the server can directly send the response to that client. Therefore MaTaDoR allows to establish the communication of legitimate users and the genuine web server.

Whenever a packet authentication fails, MaTaDoR redirects the packet to the decoy server. The flow of unauthenticated segments is shown in Figure 2b. Decoy server can also see the IP address of the initiating client. Therefore, the adversary might believe as if they are directly connected to the server, without any interceptors. Thus, unauthenticated or unwanted messages are directed to the decoy server and genuine server can continue its normal operation.

A DoS attack example in which the adversary never finishes TCP handshake to waste the resources of benign servers is shown in Figure 2c. In this scenario, like in Slowloris attacks, the adversary sends only "SYN" packets of the handshake and never responds to incoming "SYN/ACK" packets. This behavior makes serving computers to wait for an incoming connection, by allocating some of its resources to keep state. The increase of such requests makes a web server to allocate more resources and eventually the server becomes unavailable. Since MaTaDoR forwards the requests to an actual, yet decoy web server, the decoy server becomes unavailable. Thus, MaTaDoR serves its purpose by confusing the adversary either the web server is on or off. Genuine server operates normally in the meantime.

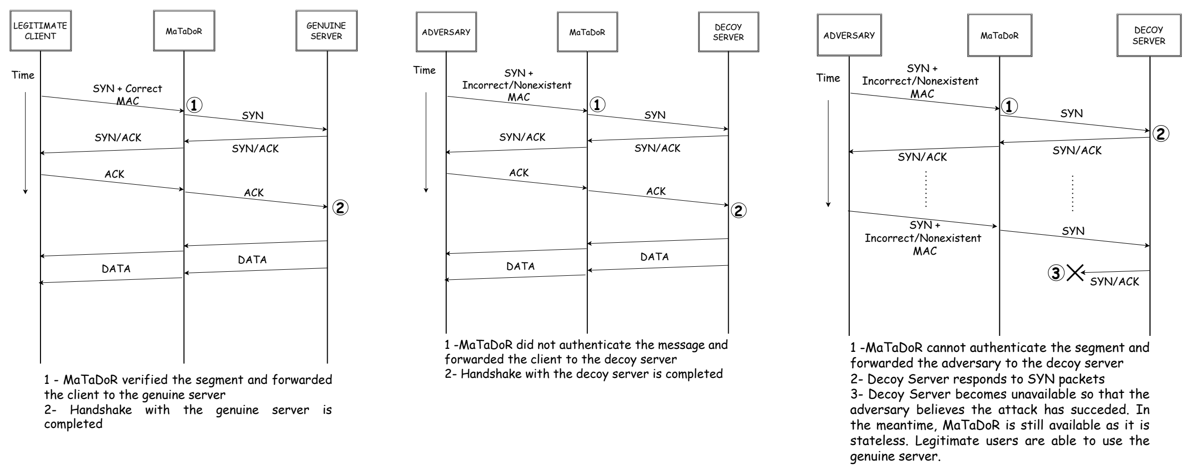
Note that, both benign and malicious traffic is handled the same from the external network's point of view. Therefore, it is not possible to detect either the genuine or decoy server is active. This attribute allows users to be unaware of that there is a protection mechanism in place.

4 METHODOLOGY

MaTaDoR's feasibility is validated by a simulation setup similar to Figure 1. A real network dump is regenerated with Scapy (Rohith et al., 2018) to build experimental traffic. *Scapy* is a packet manipulation program that enables real time header processing. Since popular operating systems like GNU/Linux, OS X and Windows do not support TCP-AO message authentication header is added at the client side with Scapy. Then, MaTaDoR handles these headers via custom iptables rules.

The original traffic data is named "SUEE1"[1], includes both wireless and wired interfaces, and collected by Lukaseder et al. (Lukaseder et al., 2018).

[1]https://github.com/vs-uulm/2017-SUEE-data-set


LEGITIMATE CLIENT
MaTaDoR
GENUINE SERVER
Time
SYN + Correct MAC
SYN
SYN/ACK
SYN/ACK
ACK
ACK
DATA
DATA
1 - MaTaDoR verified the segment and forwarded the client to the genuine server
2- Handshake with the genuine server is completed
ADVERSARY
MaTaDoR
DECOY SERVER
Time
SYN + Incorrect/Nonexistent MAC
SYN
SYN/ACK
SYN/ACK
ACK
ACK
DATA
DATA
1 -MaTaDoR did not authenticate the message and forwarded the client to the decoy server
2- Handshake with the decoy server is completed
ADVERSARY
MaTaDoR
DECOY SERVER
Time
SYN + Incorrect/Nonexistent MAC
SYN
SYN/ACK
SYN/ACK
SYN + Incorrect/Nonexistent MAC
SYN
SYN/ACK
1 -MaTaDoR cannot authenticate the segment and forwarded the adversary to the decoy server
2- Decoy Server responds to SYN packets
3- Decoy Server becomes unavailable so that the adversary believes the attack has succeded. In the meantime, MaTaDoR is still available as it is stateless. Legitimate users are able to use the genuine server.

(a) Communication flow of authenticated messages that are destined to genuine server.

(b) Communication flow of unauthenticated messages that are destined to genuine server.

(c) Communication flow of a DoS Attack.

Figure 2: Characteristic packet flows of MaTaDoR.

The traffic includes both incoming and outgoing HTTP/S packets. The malicious packets are separately generated by running *slowloris* and *slowhttptest* DoS attacks. The malicious source addresses are chosen from a pre-determined IP space during the experiments so that traffic statistics could be compiled easily at the target.

The experimental traffic uses exact packet timing, order and size of the original. Besides, the packet contents are masked and the traffic is directed to the target host either with enabled authentication or not. The included malicious traffic is also an exact copy of a pre-recorded DoS attack.

The overall traffic data includes a total of 2 089 436 packets, where 1 962 955 of them are benign requests and 126 481 of them are DoS attempts.

When the DoS attack is conducted for any of the web servers, it takes about 15 seconds to disrupt their services. It is possible to observe this by not receiving any acknowledgment messages from the servers. MaTaDoR does not alter the original packet, it only marks these packets based on the hash-based message authentication codes (HMACs) (Krawczyk et al., 1997). Then the marked packets are forwarded to genuine or decoy server depending on its authenticity. The proposed approach is suitable and compatible to be used by utilizing the optional TCP-AO header. However, it is omitted for two resasons. First, it is known that BGP frequently uses this option in between autonomous systems (ASs) and this may cause that the proposed message authenticated packets be filtered via BGP routers. This is the main reason that the provided implementation allocates another optional header area. Still, MaTaDoR could safely be used via TCP-AO header within the same AS. Second, end user devices do not support TCP-AO by default. It is more practical to utilize optional additional header.

In the final design, benign and malicious traffic are processed together, both with message authentication header and not, to measure the overhead generated by the proposed solution. In order to observe the efficiency of the mechanism and observe the effects of the attacks in the experiments, the server software chosen to be vulnerable against DoS attacks. This is a natural requirement that should be satisfied to see that unwanted or unauthenticated packets never reach the genuine server. In this expreriment, unwanted traffic is the *slowhttptest*, *slowloris* and *slowloris-ng* packets from the SUEE1 dataset. The DoS vulnerable server software is chosen as the built-in Python2 module called *SimpleHTTPServer*. As this software has limited connection capabilities, it is easy to know that if it is disrupted by simply checking its status or by observing whether it returns an acknowledgment packet.

MaTaDoR's source code is available on Github.[2]

5 RESULTS

The overhead of using authentication header and the additional redirection is measured to present the performance of the proposed solution. Built-in *time* command is utilized to measure the execution time of authentication verification. Moreover, the additional

[2]https://github.com/BerkanUfuk/MaTaDoR

load of the verification to the CPU is collected.

The experiment is conducted on a Raspberry Pi 3 Model B Plus Rev 1.3 and two regular personal computers with Intel Core i7-7700HQ CPU and 8 GB RAM. One of the computers have the web servers and named as *server computer*, and the other computer has client software and named as *client computer.* Raspberry Pi resides as the MaTaDoR in the experiments. The *server computer* has 2 virtual machines which have identical web servers constructed with python's *http.server* module. Also, these two virtual machines have the same operating system and configuration. The only difference is that they are on different interfaces and on different physical sub-nets that have the same network configuration. *client computers* also have the same operating systems and placed at the other end of MaTaDoR. The clients in the experiments are GNU/Linux virtual machines with Scapy to send packets.

Two scenarios are tested to point out the performance difference of using authentication or not. The experiment is repeated five times and results are provided as the arithmetic mean of each. Table 1 summarizes the perspective of the client computers. According to the table, the proposed solution has limited effect on the CPU load. The overall delay is noticeable but the slight increase seems acceptable.

The columns shown on the Table 1 are as follows: The *Real* column indicates the total amount of time to process the traffic during the experiment. Similarly, *User* and *Kernel* indicate the amount of time consumed in user and kernel space. *CPU* indicates the mean of additional CPU load throughout the experiment. Finally, *RTT* is the round trip time to send request from the client to the either of the servers, and receive a response. Recall that, there are 1 962 955 benign and 126 481 malicious packets exist in the regenerated pre-recorded real world traffic.

The total time spent for each benign packet is around 25 ms. This is around 7 ms for each malicious packet. This difference is due to the additional response originated from the application layer processed for benign packets. Therefore, this difference cannot be considered as an additional cost of MaTaDoR, but as the natural outcome of using the network resources.

The overhead of additional message authentication header for benign traffic is approximately 8% and the difference in the CPU load is negligible. The overhead is approximately 2% for malicious traffic. These results indicate that MaTaDoR effectively and efficiently filters out the malicious or benign traffic and the overhead is depend mostly on the packet size. The accuracy of this filtering is 100% as MaTaDoR relies on cryptography unless a noise disrupts a packet.

Table 2 represents the average CPU load and additional CPU loads as observed from the MaTaDoR. In order to measure the CPU load, the built-in *top* command is used in the MaTaDoR. As it can be seen, MaTaDoR brings up an extra 4.3% CPU cost. This is due to IPTables rules that MaTaDoR builds.

MaTaDoR also has an insignificant effect in packet delays. The average time of packet delays with MaTaDoR is 597 microseconds, while the average packet delay is 487 microseconds without MaTaDoR. The RTT delay is measured with Scapy by subtracting acknowledgment response time from packet sending time. Additionally, 90 microseconds of RTT difference seems unnoticeable by the users of a network. It could be speculated that this difference mainly roots from the additional router hop. This slight difference better demonstrates the undetectability of the proposed solution. It could be seen that the transparently MaTaDoR barely effects the network behavior. Therefore, it is very hard to notice if an additional defense mechanism is in use in the network. It could also be seen that, MaTaDoR has a very limited CPU usage

Table 3 presents a comparison of the proposed scheme with GhostMTD (Park et al., 2020). GhostMTD is chosen as its quantitative experimental results are available. Note that, GhostMTD is similar to MaTaDoR as both relies on message authenticating bit streams at the network border and they both forward invalid packets to decoy servers. However, GhostMTD requires a connection to be established to analyze incoming packets and keeps local state, so it is not scalable.

The results presented in the table are normalized as experimental setups are different. MaTaDoR introduces around 2.86 % throughput performance loss while GhostMTD decreases the performance by 3.84 %. MaTaDoR's efficiency is likely to be the result of its use of authentication via TCP header early before establishing the connections. Thus, packets are handled early without further inspection in the payload.

6 CONCLUSION

In addition to conventional network security practices, availability is a critical and expected feature. MaTaDoR chaffs away unauthenticated traffic smoothly by introducing message authentication. Therefore, it provides strict access control and availability at the same time. MaTaDoR acts as a transparent router with authentication key based filtering

Table 1: End to end performance metrics of the client computers.

Traffic	Real [s]	User [s]	Kernel [s]	CPU [%]	RTT [µs]
Benign w/o MaTaDoR	49159	827	4803	13	512
Benign w/ MaTaDoR	52647	1534	5132	14	598
Malicious w/o MaTaDoR	1332	12	186	16	462
Malicious w/ MaTaDoR	1467	32	365	17	631

Table 2: CPU load comparison.

Traffic	CPU [%]	CPU Difference [%]
w/ MaTaDoR	0.7	4.25
w/o MaTaDoR	0.7	none

Table 3: Throughput comparison of GhostMTD vs. MaTaDoR.

	GhostMTD	MaTaDoR
Loss [%]	3.84	2.86

capabilities. As a result, it is stateless and easily scalable. This feature could be helpful to interconnect two semi-trusted organizations without the necessity of a leased-line or a VPN. Moreover, such a mechanism is not directly detectable. This is useful to lure adversaries to honeynets, which is not discussed in depth in the paper and left as a future work.

It is obvious that a practical authentication key exchange will be necessary to increase the usability of the proposed mechanism. The authors aim to conduct a complexity study after further refining the provided code.

REFERENCES

Bogosyan, S., Akgul, T., and Gokasan, M. (2020). Mtd based novel scheme for bms security against can bus attacks during bev charging. In *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–7. IEEE.

Hong, J. B. and Kim, D. S. (2015). Assessing the effectiveness of moving target defenses using security models. *IEEE Transactions on Dependable and Secure Computing*, 13(2):163–177.

Jafarian, J. H., Al-Shaer, E., and Duan, Q. (2012). Openflow random host mutation: Transparent moving target defense using software defined networking. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 127–132, New York, NY, USA. Association for Computing Machinery.

Jenkins, C., Vugrin, E., Manickam, I., Troutman, N., Hazelbaker, J., Krakowiak, S., Maxwell, J., and Brown, R. (2021). Moving target defense for space systems. In *2021 IEEE Space Computing Conference (SCC)*, pages 60–71. IEEE.

Kampanakis, P., Perros, H., and Beyene, T. (2014). Sdn-based solutions for moving target defense network protection. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6.

Krawczyk, H., Bellare, M., and Canetti, R. (1997). HMAC: Keyed-Hashing for Message Authentication. Request for Comments 2104, Fremont, CA, USA: Internet Engineering Task Force.

Lukaseder, T., Maile, L., Erb, B., and Kargl, F. (2018). Sdn-assisted network-based mitigation of slow ddos attacks. In *International Conference on Security and Privacy in Communication Systems*, pages 102–121. Springer.

Luo, Y.-B., Wang, B.-S., Wang, X.-F., Zhang, B.-F., and Hu, W. (2017). Rpah: A moving target network defense mechanism naturally resists reconnaissances and attacks. *IEICE Transactions on Information and Systems*, E100.D(3):496–510.

MacFarland, D. C. and Shue, C. A. (2015). The sdn shuffle: Creating a moving-target defense using host-based software-defined networking. In *Proceedings of the Second ACM Workshop on Moving Target Defense*, MTD '15, pages 37–41, New York, NY, USA. Association for Computing Machinery.

Park, J.-G., Lee, Y., Kang, K.-W., Lee, S.-H., and Park, K.-W. (2020). Ghost-mtd: Moving target defense via protocol mutation for mission-critical cloud systems. *Energies*, 13(8).

Rivest, R. L. et al. (1998). Chaffing and winnowing: Confidentiality without encryption. *CryptoBytes (RSA laboratories)*, 4(1):12–17.

Rohith, R., Moharir, M., Shobha, G., et al. (2018). Scapy-a powerful interactive packet manipulation program. In *2018 international conference on networking, embedded and wireless systems (ICNEWS)*, pages 1–5. IEEE.

Touch, J., Mankin, A., and Bonica, R. P. (2010). The tcp authentication option. Request for Comments 5925, Fremont, CA, USA: Internet Engineering Task Force.