OpenMTD: A Framework for Efficient Network-Level MTD Evaluation

Richard Poschinger*
richard@poschinger.net
Ludwig-Maximilians-Universität München

Nils Rodday
Research Institute CODE
Universität der Bundeswehr München

Raphael Labaca-Castro
Research Institute CODE
Universität der Bundeswehr München

Gabi Dreo Rodosek
Research Institute CODE
Universität der Bundeswehr München

ABSTRACT

Moving Target Defense (MTD) represents a way of defending networked systems on different levels. It mainly focuses on shifting the different surfaces of the protected environment. Existing approaches studied on network-level are Port Hopping (PH), which shifts ports, and Network Address Shuffling (NAS), which steadily alters the network addresses of hosts. As a result, the formerly static attack surface now behaves dynamically whilst the relationship of ports to services and network addresses to hosts can be changed. Most MTD approaches have only been evaluated theoretically and comparisons are still lacking. Hence, based on existing results, it is not possible to contrast implementations like PH and NAS in terms of security and network performance. Finally, implementation details are usually not shared publicly. To mitigate these shortcomings, we developed a hybrid platform that evaluates such techniques and reimplemented PH and NAS with additional features such as connection tracker with fingerprinting service and a honeypot module, which is helpful to bypass attackers attempts. We created a common software platform to integrate approaches using the same gateway components and providing graphic network usability. The environment, named OpenMTD, has been open-sourced and works in a modular fashion allowing for easy extensions and future developments. We show that common attacks, starting with a reconnaissance phase were not able to successfully reach vulnerable hosts that are part of the OpenMTD-protected network. A new worm has been developed to spread across the network and the propagation paths showed that OpenMTD can lay the ground for extending protection mechanisms against self-propagating threats.

CCS CONCEPTS

• **Networks** → **Network security**; • **Security and privacy** → **Network security**.

*Work was done during a research visit at the Research Institute CODE.

MTD'20, November 9, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8085-0/20/11…$15.00
https://doi.org/10.1145/3411496.3421223

KEYWORDS

openmtd, moving target defense, network security, port hopping, network address shuffling

ACM Reference Format:
Richard Poschinger, Nils Rodday, Raphael Labaca-Castro, and Gabi Dreo Rodosek. 2020. OpenMTD: A Framework for Efficient Network-Level MTD Evaluation. In *7th ACM Workshop on Moving Target Defense (MTD'20), November 9, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3411496.3421223

1 INTRODUCTION

With the number of cyber attacks increasing dramatically over the past years, static defense mechanisms are not sufficient anymore to protect networks in an adequate way. Instead, more agile and dynamic network security approaches need to be evaluated and deployed. MTD satisfies those requirements by continuously shifting the attack surface, making previously static parameters appear to be dynamic. Well-known approaches are Network Address Shuffling (NAS) and Port Hopping (PH). While the former keeps shuffling Internet Protocol (IP)-addresses, the latter alters the ports used for active transmissions. Implementations can be complex and network operators are hesistant towards the introduction of moving parameters into their networks, as debugging and troubleshooting becomes cumbersome. Hence, MTD approaches should be carefully designed and evaluated against each other. However, almost every previous publication relied on a different evaluation environment, specifically tailored towards the evaluation of a single proposal and not accessible to fellow researcher, leaving little room for objective comparison and evaluation within the same set of parameters. Therefore, we identified the need of the MTD research community for a common framework to perform comparisons of different proposals. This paper presents OpenMTD, a modular environment designed to evaluate MTD methods, based on the well-known network simulation tool GNS3 [11]. It is easily deployed and ready to reproduce existing methods, while flexible enough to quickly incorporate new concepts. We use our newly proposed framework to evaluate two major concepts, PH and NAS in order to provide a starting point. Moreover, we developed several extensions, suitable for both approaches, including a connection tracking module and a honey pot module to satisfy requirements of modern networks. Since artifacts of previously published concepts are not publicly available, we contribute to the current state-of-the-art by also fully releasing our implementations of the aforementioned MTD approaches.

A cyber-attack usually starts with the identification of targets, also known as reconnaissance [19]. In a static network, an attacker has nearly unlimited time to discover services and vulnerabilities. Once identified, the attacker is able to use that knowledge almost indefinitely as components of networks are not changing frequently. We show that NAS and PH significantly reduce the exposure of internal systems towards external threats, making it much harder for an adversary to find vulnerable targets within the protected infrastructure. Even if such targets are identified, the resulting knowledge is only exploitable for a very short period of time as the MTD-protected network would continue to change its parameters. Our newly developed NAS connection tracking module allows for continuous connections although subnets and addresses are being exchanged. External clients receive a different virtual IP and port to interact with the same service, while old connections are maintained to retain a high level of service quality.

We evaluate the security of each approach by performing several networks scans from an attacker's perspective using different NAS and PH parameters and measuring the obtained insider knowledge. Since most attack mechanisms are based on scanning tools, e.g. NMap [21], they rely on the assumption that networks are static. We additionally wrote our own attack script that would adapt to quickly changing networks, as a sophisticated state-sponsored adversary might be able to, and constructed a worm based on these observations that would infect vulnerable systems and quickly spread throughout a network. We measure the number of infected hosts for each experiment run to make judgments about the security level of evaluated approaches.

In summary, this work presents the following contributions: (i) Development of OpenMTD, a modular MTD evaluation platform that allows for ease of reproducibility and consists of a virtualized testbed and a software framework; (ii) implementation of PH and NAS in a hybrid approach that leaves it up to the operator's discretion which features to use; and (iii) further development of additional modules, e.g. connection tracking and honeypots, to extend current approaches. All of those parts have been made publicly accessible. Moreover, (iv) we evaluate PH, NAS, and our hybrid approach within OpenMTD against each other and lastly (v) the development of a proof-of-concept worm, which performs very aggressive scanning and immediate exploitation afterwards to penetrate the additional layers of defense.

The remainder of this paper is organized as follows: in Section 2, we provide the background and a literature review necessary to introduce the environment and the hybrid approach in Section 3. Section 4 is the evaluation of approaches in terms of performance and security using the metrics defined in the previous section. Section 5 concludes the work.

2 RELATED WORK

In order to perform evaluations with our OpenMTD environment, we are going to, first, develop and implement network-level MTD approaches based on the literature while extending some of the capabilities of the proposed techniques. In general, two MTD approaches, Network Address Shuffling and Port Hopping, have been most prominent in this area of research during the past years. We

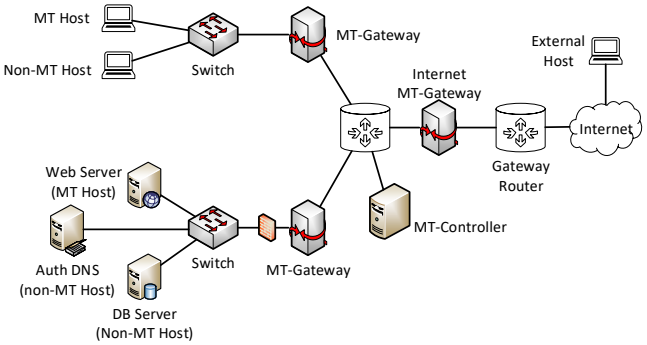


Figure 1: Random Host Mutation, cf. [1]

are going to provide an overview over these two, including additional variants proposed in the literature.

Network Address Shuffling. IP addresses are used for the identification of sender and destination within the Internet. The Internet Protocol relies on lower layers to provide switching capabilities, and subnetting, a way to divide the address space into smaller portions. This is helpful for several technical and administrative reasons [23, 31]. As subnets provide smaller address ranges compared to the overall available space by definition, they can more easily be scanned for active hosts. NAS provides a means to dynamically alter network addresses. It changes the mapping between address and host, which historically remains static. Addressing information invalidates over time as addresses keep changing. Once acquired insider knowledge can therefore only be used during a short amount of time. Server systems can use a real IP (rIP) which is changed by the NAS system or use an additional virtual IP (vIP). The usage of a vIP results in a constant rIP.

Jafarian et al. [14] introduced the Open Flow Random Host Mutation (OF-RHM). It is based on the OpenFlow protocol and requires Software Defined Networking (SDN). One address range is assigned per host. The vIP is chosen from the assigned address ranges with uniform probability associated with each vIP. The system uses different components for the address translations (Moving Target Gateway (MT-Gateway)) and the coordination of the mutation (Moving Target Controller (MT-Controller)). The controller defines new sets of vIPs, coordinates the OpenFlow switches and handles Domain Name Service (DNS) updates.

Al-Shaer et al. [1] propose RHM, which is a modified version of OF-RHM and does not require SDN. It also uses vIPs as well, which get assigned in a two-phase allocation mechanism. Figure 1 shows the structure of a RHM network. A MT-Controller assigns a subnet to a host during Low Frequency Mutation (LFM) and sends this information to the gateway, which is responsible for the vIP $\rightarrow$ rIP translation of the host. The MT-Gateway chooses an IP address of the given subnet to the protected host if High Frequency Mutation (HFM) is triggered. They also translate DNS responses and replace the rIP with the vIP.

Antonatos et al. [2] propose Network Address Space Randomization (NASR), which changes the rIP while the host is inactive. This is done using the Dynamic Host Configuration Protocol (DHCP). The implementation has to be altered to enable it to keep track of

TCP connections and avoid address changes while connections are still active. The second new module, which has been added to the DHCP server is called Service Fingerprinting. It keeps track of every connection and decides whether a connection failure is tolerable. If it is tolerable, the DHCP server will change the address even if alive connections exist. The client can reestablish the connection via DNS address resolution. Simulations have been used to test the worm propagation in an NASR-protected network.

Jafar et al. [15] proposed a "Spatio-temporal Address Mutation" approach which also uses DNS requests. Only authorised administrators are allowed to reach hosts by the real address. These temporal addresses (ephemeral IPs) are generated using the source (requestor) identity and the time. The rIP remains unchanged. The mutation can be executed using two strategies. The random mutation strategies chooses an address from the unused address space randomly. If an attacker tries to probe the network, the deceptive mutation will be activated. This strategy tries to redirect new probes from the attacker to an already visited host.

Fraunholz et al. [8] proposed an approach to increase the complexity of reconnaissance and prevent fingerprinting methods. A hash-based address mutation technique is combined with different stack randomization mechanisms, e.g. TCP values can be tampered. To prevent device identification, the approach is able to randomize the originally hardware-bound MAC-Address Additionally, decoy systems are part of the network and are deployed to unused adresses based on automatic network cluster analyses. The approach has been implemented in Python using new kernel modules.

Port Hopping. Another significant approach is PH, which is based on the alternation of ports used by the transport layer protocols User Datagram Protocol (UDP) and Transmission Control Protocol (TCP). Both protocols use ports to identify the processes that handle outgoing or incoming data. For TCP connections source and destination ports are required to identify an individual connection. UDP has a header field for both ports as well, but, by default only requires a destination port, as it is a connection-less protocol, which does not support many features of connection-oriented protocols. The ports are represented by two octets in the headers of both protocols [25, 26].

Yue-Bin et al. [20] elaborate in their work on modeling the effectiveness of port hopping approaches: To obfuscate the Well-Known (1-1023) or Registered Ports (1024-49151) which are normally used by server systems to provide services, always reachable under the same destination port, PH constantly maps the selected ports of the services to generated ports. The generated ports are chosen from a port pool that is available to the administrator. The resulting generated port is also called vPort. The original service port is called rPort and is blocked once PH is active. Most of the approaches use a function to generate the vPort based on the rPort and vice versa. This function can additionally rely on a pre-shared key. They show that PH can limit the likelihood of an successful attack.

Badishi et al. [3] developed a DoS-resistant protocol. It is similar to its connection-less UDP and IP relatives and makes use of a pre-shared key. The proposed method relies on acknowledgements, which eliminates the problem of synchronisation. They use a pseudo-random function to generate the next port during a PH sequence. A counter is used to determine which port should be open or used as the destination port. A simulator for this approach has been developed by Hari et al [12]. They were able to prove that the success rate was still more than 90% during an attack with about 10.000 packets.

Lee et al. [18] use a time slot dependent function, which also uses a pre-shared Key. Packets with invalid port numbers will be filtered out. Overlapping time slots solve the problem of time synchronization errors and transmission delays. Symmetric key mechanisms or public key mechanisms can be used as key management solutions. The approach is incompatible with UDP, but it can be adapted for TCP with limitations. These limitations exist due to the initial port resolution at the beginning of a connection.

Hybrid approaches.

Dunlop et al. [7] developed Moving Target IPv6 Defense (MT6D), which uses a rotation of source and destination address and the transport layer address (Port). It focuses on version 6 of the Internet Protocol. The next interface identifier of the address is generated based on a hash of the current identifier, a shared key and the current timestamp. Two addresses are used concurrently to prevent interruptions of the connection. The host can define a specified port range for PH or use the unused bits of the hash to calculate a vPort. Tunnelling can be implemented and encrypted.

Kewley et al. [17] use pre-shared keys for Dynamic Network Address Translation (DYNAT). It requires a priori knowledge about the shuffling algorithm and the pre-shared key. The vPort and vIP are generated based on the rPort, the rIP, and the key. Similar to other approaches, changes are required on client and server-side. It is therefore separated from the host and transport protocol independent.

Finally, there are also modular approaches, which do not directly fit into the defined categories but are worth to be mentioned. Fraunholz et al. [9] also developed Cloxy, which is a modular Context-aware Deception-as-a-Service Reverse Proxy for Web Services. It is able to perform MTD on web application parameters. It can be adapted to the use case of different applications. Examples that are part of the prototypic implementation are version trickery, injection prevention, or cookie scrambling. The security of the implementation of the proxy has been proven and is able to deploy deceptive elements.

Furthermore, Connell et al. [5] presented a framework for the qualification of MTD techniques. Hence it is possible to determine the effectiveness of the techniques based on the probability of the attackers' success. The framework focuses on the reconnaissance phase. It is separated into four different layers which address the protected services, their weaknesses, the required knowledge and the MTD techniques. These layers are formally defined in a mathematical model which results in a probabilistic computation from the MTD to the service layer.

3 OPENMTD

Utilizing the insights from pevious work, we next elaborate on how we designed OpenMTD. We supply the image of the virtualized testbed to be readily deployed and the software platform for fellow researchers [29]. First we introduce our requirements, then explain

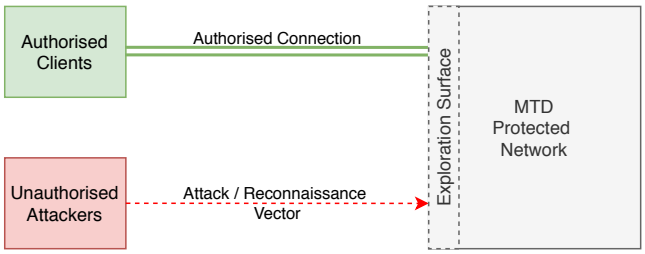


Figure 2: Evaluation of the MTD Approaches

our MTD modules, show our test environment and finally present our software platform.

Requirements. Due to the high number of different approaches, which show considerable differences in their focus and technical background, it is necessary to define metrics to select or combine approaches for the evaluation. We define the following requirements: Ease of implementation in production networks; Not dependent on SDN technology.

The first requirement is that the selected approach can be implemented in a production network without or with as few changes to the system components as possible. Above all, no elementary changes to the specification of fundamental network components or standards should be necessary. Thus all approaches, which included changes to protocols of the internet and transport layer have been excluded from further selection processes. To keep the number of changes in a production network as low as possible, the solution should not require additional software or hardware components for every involved client or server. Additionally, the used technologies should, as far as possible, preserve or use the existing infrastructure to make the transformation of a non-MTD-protected network to an MTD-protected network as simple as possible. The MTD approach should not interrupt existing administration techniques and communication. To make the selected approach even more flexible, IPv4 and IPv6 should be supported.

Second, to be able to use or evaluate the approaches in existing networks, the usage of SDN should not be a requirement or fundamental part of the selected solution. Currently, the SDN market is growing rapidly, but still, about 77 % of the data centres are not using SDN or Network Function Virtualization (NFV)[4]. Therefore, we do not want to exclude traditional networks and care for compatibility.

The evaluation measures authorized connections to the protected sector or use unauthorized attack or reconnaissance vectors as shown in Figure 2. Thus, the main focus of all approaches is the behaviour on the attack surface. Internal mechanisms, like administrator access mechanisms, which are required for the administration of the networks are not to be prioritized if they have no practical impact on the security or performance. Based on this specification, the selected approaches can also be adapted and reduced to their required features.

Network Address Shuffling Module. Network Address Space Randomization and Random Host Mutation are approaches based on NAS that are worth considering. In both approaches DNS is used to receive the address of the server. NASR uses DHCP to change the rIP of the server periodically. This might be a simple solution for

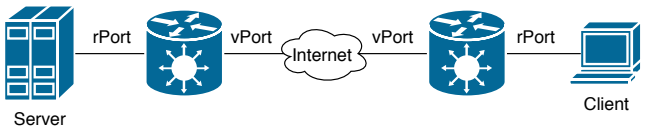


Figure 3: Modified Version of DYNAT

the generation of new addresses, but it makes the internal administration much harder. On the other hand, RHM solves this problem and uses vIP to address the server publicly. Therefore, gateways are required to perform the address translation. This can result in additional implementation and configuration effort. Nevertheless, RHM uses it as an advantage. Instead of the distribution of just one defined subnet via DHCP, RHM can shuffle in much larger address spaces. Several subnets or one large subnet can be combined and used across the infrastructure. Thus the predictability of addresses is much lower, as it would be, if the attacker could gain knowledge about which part of a network or which data centre is using which subnet.

Nevertheless, some advantages of NASR will still be useful, for instance the service fingerprinting approach as it can determine when a connection should be interrupted by a shuffling process. Therefore, the admin is required to get information about the stability of the network protocols. RHM uses connection tracking instead. Connection tracking can allow existing connections to continue even if the vIP the connection is using has changed. The technique has been integrated in the MT-Controller to support connection continuation after LFM has been triggered. Fingerprinting has been integrated using the connection tracking module and can be used concurrently for different protocols.

In the RHM environment, attackers can directly determine if the destination address is currently valid and start additional reconnaissance activities or the attack. Furthermore, it is not possible to gather more information about the activities of an attacker besides the log data containing refused packets. [35] solves these problems using honeypots, which are part of the MT-network. Every packet that contains an invalid destination address is automatically forwarded to the honeypot. Thus, honeypot support has been integrated in the NAS module of MT-Gateway and MT-Controller [1, 2]. In summary, the following improvement are part of the NAS module: Combined connection tracking with service finterprinting, Connection continuation support in the MT-Controller, Honeypot support in MT-Controller and MT-Gateway.

Port Hopping Module. As no approach suffices the requirements, DYNAT was most fitting. It is per nature a hybrid approach comprising PH and NAS, whereas only the PH module will be used. The infrastructure consists of a client and a server gateway, which can be separated from the host. In the original proposal, the DYNAT client gateway is a software on the client host. It could also technically be used in a separate component, e.g. another gateway, as shown in Figure 3. Thus, as requested in the requirements, no client or server modifications are necessary to implement this approach but the approach has to be slightly adapted as only PH functionalities can be used.

The DYNAT approach translates parameters in the TCP and the IP headers. Both gateways require an initial secret value. A cryptographic algorithm performs the address translation (vPort → rPort,

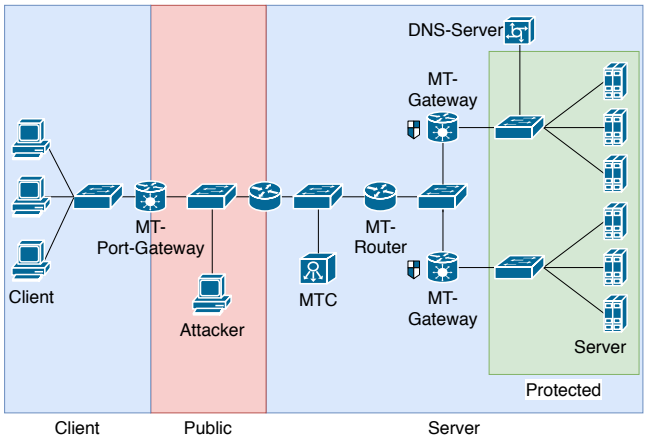


Figure 4: Testbed Topology

vIP → rIP and vise versa). The gateway recalculates the checksums afterwards. A time-based mechanism changes the secret periodically. Hence the results of port scans cannot be used continuously. The DYNAT gateways are synchronised by wall-clock time.

Figure 3 shows the modified version of the DYNAT approach. Both gateways have been arranged symmetrically. The gateways mark the borders between virtual and real ports. The internet could be replaced with any network that is accessible for a potential attacker. It could also be a part of a company network, which can be reached by employees with a lower security classifications or lower group membership rights [17, 20]. It was possible to speed up the PH module, by integrating a cache for the secure hashes. Thus, the hashes don't need to be recomputed laboriously for the same tuple of the time interval, the source rIP and the pre-shared key. In summary, the following improvement are part of the PH module: Hash-cache for higher performance and client gateway separated from the host.

Environment. A simplified version of our test network, representing the general platform in which previously introduced modules can be enabled or disabled, is depicted in Figure 4. The green box marks the protected area. These hosts can only be reached using virtual addresses or ports. The DNS-Server is excluded from the protected network, as it needs to be addressed statically. Therefore, the gateway needs to support both, static addresses and address translation for protected hosts. The MT-Gateway connects the protected networks to the rest of the network. The Moving Target Router (MT-Router) not only performs static routing to the gateways, it also has to provide the dynamic virtual routes. The MT-Controller is part of the server network as well and can directly communicate with the MT-Router and the MT-Gateways. The router in the middle only provides static routing functionality and marks the border to the public network. Attackers can be simulated, and security tests can be launched from hosts in the public part of the network. The blue client-side contains hosts, which can communicate via a secure port hopping protected channel. An additional gateway is part of the client network, to perform the port translation between client and server networks. Even if it contains the same implementation of the gateway software, port hopping functionality has to be activated. To distinguish it from other gateways it is called Moving Target Port Gateway (MT-Port-Gateway).

The environment needs to support all aspects of the required tests, including the required protocols and needs to show realistic networking behavior. Furthermore, it has to support all required protocols and needs to show realistic networking behavior. This is not only required for performance tests, it would also impact security analyses, as they rely on different protocols. To perform security tests, it must be possible to use real network security or penetration testing tools in the network. Hence network simulations like OMNET++ or ns-3 cannot be taken into account. A virtualised environment has been preferred over a hardware implementation to limit the financial and administration effort. Frameworks like ns-3 and Mininet fell prey to our previously mentioned exclusion of SDN technology. GNS3 is highly flexible and can perform most of the networking tasks [24]. It supports different types of virtualized devices and can be easily administrated using a GUI. The GUI can be installed remote from the actual environment, which can run on another computational platform and thus can use e.g. cloud computing resources.

In order to obtain a running setup, a fellow researcher would have to follow these simple steps, which are more closely described in the HowTo provided with the image: (i) Download the VM image (server) (ii) download and install the GUI software (client) (iii) adapt network adapter configuration of VM (iv) setup deployment configuration (client-server connection).

The test environment contains three host groups, each containing 10 hosts, which have been used for worm propagation tests. Two server-side and one client-side gateway perform MTD, are controlled by an MTC and can forward traffic to two honeypot servers. A DNS server has the features required to resolve domain names. Many other hosts provide resources for performance and security testing. Configurations are distributed via a deployment network.

Software. The MT-Gateway provides translation capabilities for both NAS and PH and has been implemented in Python. A modular architecture has been created which is able to configure and set up all modules that are responsible for packet manipulation. It provides the technical capabilities to receive packets, alter them, and reinject it into the network stack using fnfqueue (a faster version of NetfilterQueue) [33]. Each module can subscribe to the information of certain protocols. Among a few other Python libraries Scapy is used to perform packet analysis and manipulation. It supports both IPv4 and IPv6. The platform can maintain a certain procedure to ensure compatability between different modules and support performance optimizations like the parallel execution of connection tracking modules. As the MT-Controller has to control the MT-Gateways and does not share any address translation and packet manipulation functionality, it has been implemented separately.

New modules can easily be added by following these steps (again described more closely in the HowTo shipped with the VM): (i) Create a new class (tracker or manipulating), (ii) define the necessary protocols, (iii) add the new module to the packet manipulation sequence, (iv) add to the setting file.

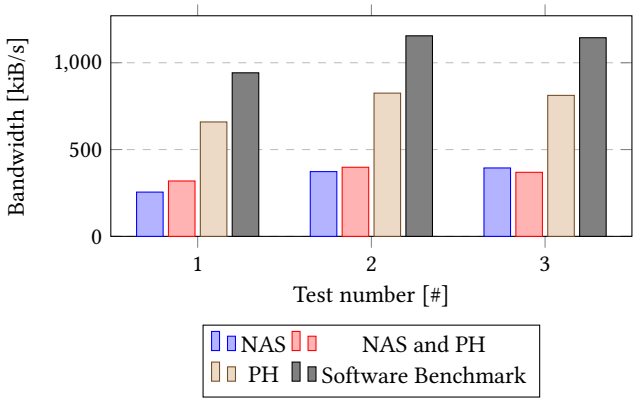


Figure 5: NAS and PH - Curl Download Results

4 EVALUATION

The evaluation focuses on security and performance aspects of the studied approaches. Specifically, security evaluates the reconnaissance surface and the consequences of its behaviour while performance measures basic key figures and the behaviour of high-level technologies and protocols.

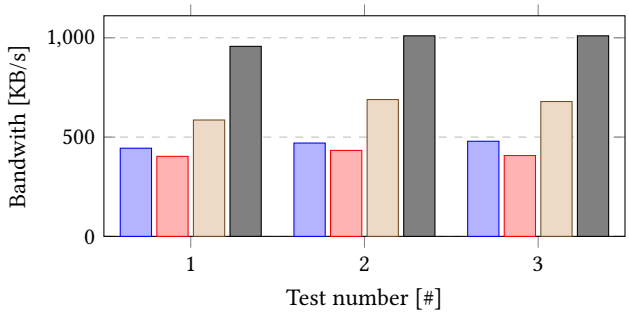
4.1 Performance

The performance of the network can be measured in different ways. Based on the categories of Lee et. al. [16] the following metrics have been selected:

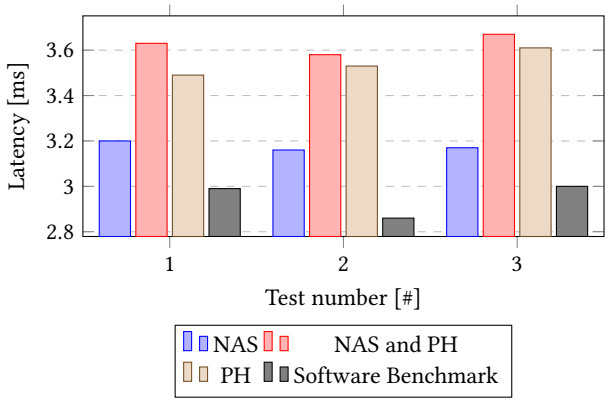
(1) Transmission speed (Utilization),
(2) Latency (Delay),
(3) Loss and Availability

Transmission speed. The speed is measured from the authorized client network towards the server network. In this case, TCP-based methods are employed to evaluate the speed of data transmission, since only TCP connections continue to work after the shuffling process. Curl [6] has been used to evaluate the performance of downloads via HTTP from an Apache webserver. Experiments are launched separately to trigger DNS resolution. Otherwise, it could exceed the LFM sliding window of NAS. On the webserver, a 50 MB file *testfile* has been created, which is then transferred for measurement purposes. The default configuration is the following: For NAS, 30 seconds HFM and 120 seconds LFM cycles are set with connection tracking (continue all mode) and the sliding window size is two LFM cycles. Whereas for PH, the hopping period is 20 seconds. Finally, DNS TTL has been set to three seconds. The experiments are performed using NAS and PH separately and results are combined in the end. The performance with (platform benchmark) and without (network benchmark) the software platform have been measured during the benchmark tests. As shown in Figure 5, if the PH module is used, the speed decreases to around 800 kiB/s. Compared to NAS the PH module is much more efficient whereas the combination of NAS and PH together performs similar to using NAS alone.

In addition to HTTP, pure TCP connection bandwidth experiments have also been conducted. For instance, Qperf can measure bandwidth and latency between two hosts using different protocols like TCP, UDP and RDMA. Also, the TCP one-way bandwidth test



(a) NAS and PH - QPerf Bandwith Results



(b) NAS and PH - QPerf Latency Results

Figure 6: NAS and PH - QPerf Test Results

is used, which lasts 120 seconds and the configuration of the MT-Gateways is not changed. [10] Figure 6a points out that achieved bandwidth is lower compared to Curl. Yet, they show the same relative correlation between the different MTD modules and the benchmark. This time, the combination of NAS and PH is clearly less efficient than the usage of NAS alone.

Latency. Qperf has also been used to examine the latency of the connection to the host, which refers to the time a packet needs to be transferred across the network. It has no impact on the network throughput of UDP, but it affects TCP connections. A higher latency influences the congestion control of TCP negatively. The delay of acknowledgements will slow down the increase of the congestion control windows [30].

The results of Figure 6b show that the NAS has lower latency times than PH. Additionally, the results indicate that the negative impact on the TCP congestion control is limited. The high PH latency could be a result of the high computational effort of the initial secure hash calculation. Based on the results, the latency of PH seems to be rising with each test run. However, we conducted additional experiment which did not confirm this trend. It is therefore judged as an artifact.

For NAS, domain resolution is performed before the connection starts. Thus, the latency is important to the whole connection performance until the DNS resource record arrives. Therefore, experiments are performed to measure the latency of DNS resolution.

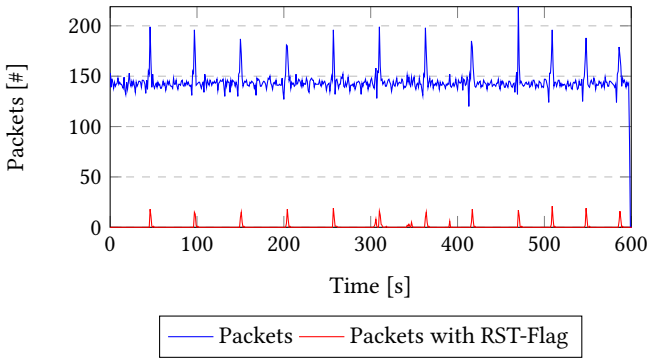


Figure 7: NAS - Wireshark HTTP Test Analysis (All Packets)

Dig has been used for this, a tool capable of performing domain resolution [13]. Our results, that we omitted in more detail for brevity, show that the latency for DNS resolution has at least doubled.

Loss and Availability. Given that both approaches, NAS and PH, change identifiers periodically this behaviour may result in instabilities when a connection is established. Furthermore, NAS and PH have different weaknesses in this regard and need to be evaluated separately with different setups.

For NAS the implemented approach combines different techniques, which can keep connections alive. The MT-Controller keeps old subnets configured at the router and the MT-Gateways can allow old connections to continue by performing address translations for outdated vIP → rIP mappings. Additionally, the connections can be analysed and the shuffling process can be postponed. For this reason no packet loss or connection interrupts were measured during several experiments when these techniques were activated.

To evaluate NAS in the network ApacheBench is used, which is a benchmark for Apache servers and can show how many requests it is able to serve [32]. In this case, 1000 requests are launched in 100 concurrent threats, which take on average 18 seconds to run. During multiple test runs, all requests have been successfully completed.

The NAS LFM procedure consists of several processes running on different hosts. This may lead to a temporary inconsistency of different vIP and mapping configurations, which can affect connection attempts. For instance, the MT-Gateway responsible for the DNS server can return outdated vIPs, while they are not reachable anymore via the MT-Gateway of the host subnet. Additionally, the timespan between DNS response and connection attempt can also result in a connection to an outdated vIP.

To evaluate the impact of this behaviour, a longer experiment was performed with connection attempts. Every 0.2 seconds, a new request to a website is launched. The results are evaluated using Wireshark as observed in Figure 7. For the HTTP requests, Curl is used like in the previous section. During the examination, 21 different IPv4 and 33 IPv6 endpoints have been recorded. In total about 12000 HTTP packets have been sent and the server never returned an HTTP error code. Thus, it was able to handle all requests successfully. Figure 7 depicts the number of packets sent during the test. The blue line shows all the HTTP packets whilst the red the number of TCP resets due to connection errors. As the LFM shuffling process is started every 120 seconds, the number

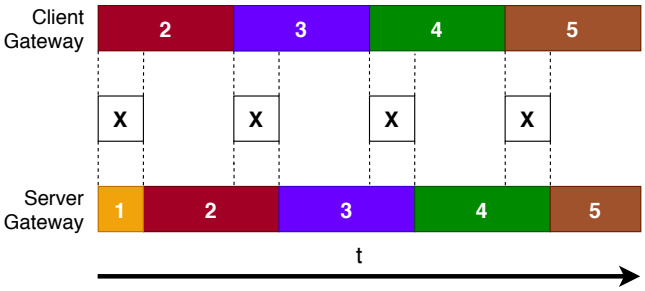


Figure 8: PH - Time Interval

of resets increases in these range. This returns short transmission peaks, which are probably caused by retransmitted packets. As only one host is performing the HTTP requests, it needs to perform the DNS resolution after the resource record has been deleted from the cache which can be attributed to the DNS time-to-live value. This behaviour can be observed, as the blue line shows little drops if the client has a valid DNS entry.

The PH approach is limited to only two modules on two hosts, which need to interact. No synchronisation is required between both gateways, as they are able to compute the port translation independently. The port translation is based on the current time and does not provide any time synchronisation functionalities. Hence, both hosts use their local timestamp that has been received using the Network Time Protocol (NTP). As the local time can vary, a difference between both PH gateways can occur. Figure 8 shows how deviations of the time synchronisation can result in timespans with an invalid state. Every coloured box represents a time interval that has been calculated by the local timestamp of the client and the server gateway. In this example, as timestamps constantly vary, both gateways cannot establish a valid communication between time intervals. These invalid timespans are visualised by white boxes between both gateway intervals. Moreover, a valid port translation can only be calculated two-thirds of the time.

The resulting behaviour can be seen in Figure 10 where both gateways are currently in inconsistent states. They use different timespans due to asynchronous timestamps. The client wants to send a packet to the port 80 of the server. The destination port gets translated at the first gateway using the time-interval 8. The second gateway tries to recalculate the rPort but uses a different

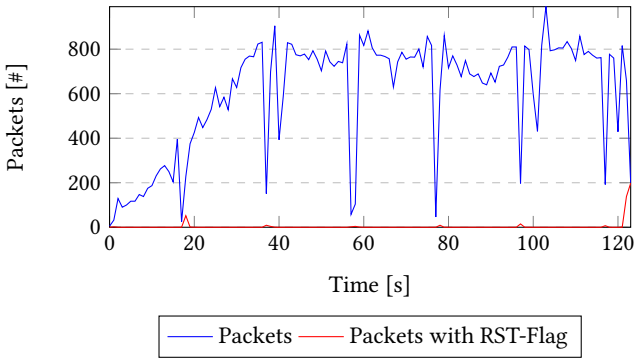


Figure 9: PH - Time Inaccuracy Bandwidth Test (Randomization Values = 300 ms)

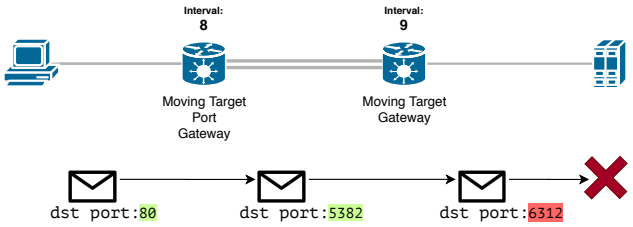


Figure 10: PH - Failed Packet Transmission

time-interval. Thus, instead of the original destination rPort 80, an invalid rPort is calculated, and the packet will not be successfully delivered. This effect can also occur due to the latency between both gateways.

To examine the impact of this behaviour, the port hopping modul has been adjusted. We randomly add or subtract time variance to the timestamp. The maximum positive and negative deviation is called *randomization value*. As both gateways use this adjusted module, the maximum derivation across the network is two times the randomization value. Qperf is used to establish a continuous connection for two minutes. Figure 9 shows that during experiments with the highest randomization value, the throughput (amount of packets per seconds) on the blue line drops every 20 seconds and thus corresponds with the PH interval. Simultaneously with these speed drops, the number of resets rises, as packets cannot be sent to the correct destination port.

A variety of randomization values have been compared indicating that the latency of the network between the gateways can be neglected as it is relatively small. The results exceed the expected values by far, since NTP can provide "accuracies generally in the range of 0.1 ms with fast LANs and computers and up to a few tens of milliseconds in the intercontinental Internet" [22]. Figure 11 shows that it has no impact on the resulting bandwidth.

During additional tests the compatibility of modern webapplications and their backend calls to NAS has been proven. Requests to NAS-protected backends did not cause any problems in webapplication during vIP changes.

4.2 Security

The main focus of the MTD modules NAS and PH is to improve the security of given networks on the exploration surface. The impact will be evaluated twofold: First, by applying reconnaissance methods to the network, namely network scans. Second, using the following real and experimental cyber-threats for the evaluation:

(1) Armitage with Metasploit
(2) Mass and optimized exploit with Metasploit
(3) PoC worm

4.2.1 Network Scans. During the preparation of attacks, it is necessary to gain information about running hosts and the services the host provides. Thus, the initial reconnaissance procedure can be divided into two aspects: finding active hosts and looking for active services. In order to find the hosts, the attacker may initially perform ping scans, for instance, using Nmap. It combines the standard ping scan, an ICMP echo request, with a TCP SYN to HTTP (port 80), TCP ACK to HTTPS (port 443) and an ICMP timestamp

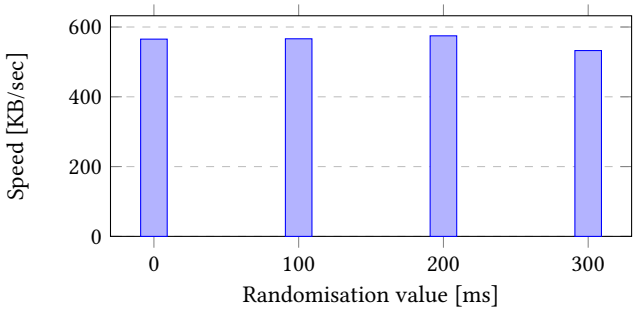


Figure 11: PH - Time Inaccuracy Bandwidth

request [21]. Hence, the Apache servers, in the test environment are easy to detect.

The Nmap scans originated from outside of the MT client network on an attacker-network host. Since connectivity to predefined standard ports is needed to perform these kinds of scans, PH is deactivated in the network, otherwise it would replace the ports with vPorts. For that reason, the evaluation of host scans focuses uniquely on the NAS approach as ports remain unchanged. The Nmap insane mode is used to scan big virtual subnets as fast as possible. During the experiments, just the virtual subnets are scanned, assuming that the attacker has already gained knowledge about the network structure. Still, the test took over 20 minutes scanning the whole subnet. Network scans of the whole server network are also performed and showed no difference in the results. In each test, the Nmap scanner was able to find several active hosts. In total eight Apache hosts were part of the server network and 48 virtual subnets could be selected by the MT-Controller. The detected subnets have been analyzed for five different tests under the same conditions as observed on Figure 12. As each virtual subnet is just attached to one host at one point in time, the subnets should not be detected twice. The blue bar indicates how many subnets and hosts have been found in total. A few subnets have been found multiple times, which can be an indicator for hosts that have been found multiple times during their movement through virtual subnets. Additionally, hosts found in different subnets could have been also detected several times, due to the changes of the virtual subnets (LFM) over the timespan of the scan process.

As previously indicated, in addition to host detection, services have also been analyzed. If just popular ports are scanned, the results show the same behaviour as without using NAS, despite the negative effects on host detection. Only if a full port scan is performed, the NAS shuffle cycle is faster than the port scan. Therefore the results become invalid. If only PH is used, it is not possible to detect open ports with the fast scan mode. Even if all ports are scanned, no results are detected. Only a few ports are open, as only Apache has been installed as a server. Since administrators try to limit the number of open ports, this can be considered a realistic setup.

4.2.2 Cyber-threats. NAS and PH are not able to limit the effects of attacks. However, some kind of threats include techniques for reconnaissance as well. These include different kinds of automated attacks and worms, which can spread across the network independently.

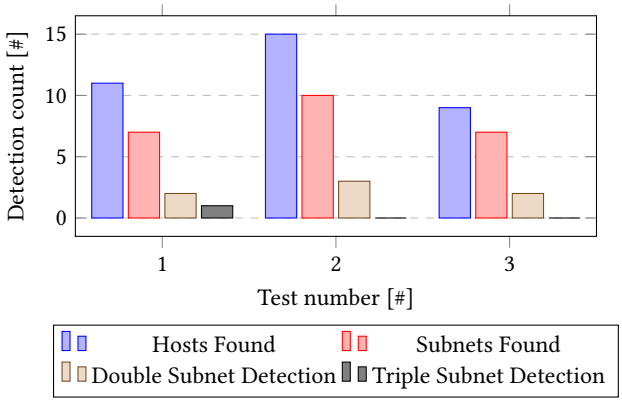


Figure 12: NAS - NMap Ping Scan Test Results

Metasploit: As previous network scans showed the effects of NAS and PH on the network security under the precondition that attacks originate from just one static location, more advanced experiments are necessary. These should be able to show the effects of additional processes in an attack procedure. Vulnerable machines were deployed in the environment, to replicate a realistic network. To be able to use realistic attacks, the Metasploit framework was used as it provides exploit management and can deliver different kinds of payloads. A host with several vulnerabilities has been set up which uses a Metasploitable Docker container.

Armitage provides an easy-to-use user interface for Metasploit and full functionality for attack management. The exploitation feature of Armitage (Hail Mary) has been used to automatically scan networks and start exploits. During experiments with PH, it was not possible to identify any vulnerabilities, as rPorts could not be identified. Additional experiments were carried out while the NAS system protected the server network. The integrated network scanners were able to identify some of the hosts but Armitage was not able to identify exploits or even launch attacks, as the vIP had already changed before the port scans could be started. As it was not possible to use the Armitage Hail Mary tool on PH and NAS protected networks, more advanced methods are necessary to evaluate the protected network more efficiently. The cause for the failed vulnerability detection was that it launched with a delay after all the hosts had been identified and the subnet was scanned completely. The mass_exploit script provides a fully automated way to perform a Hail Mary attack. As the mass_exploit script is missing host detection, it is possible to apply it to PH systems only [27]. Without PH, it was able to identify vulnerabilities and exploit them. While PH was activated, it was not possible to identify any vulnerabilities. To add host discovery functionality and directly launch the attacks after hosts have been found, an improved mass exploitation script has been created, which is able to activate the mass_exploit as soon as one host has been found. Additionally, it is possible to preconfigure exploits and set all the necessary parameters. When the optimized Hail Mary attack is launched directly after a host is found, during NAS usage, it was able to find targets on the host. However, was is not possible to launch the attack and get a remote shell. The Mass Exploit was not able to detect vulnerabilities fast enough and had to be improved manually. To increase the effectiveness of the script, the exploitation can be directly started after a host has been found and the scanning process has been parallelized. This is the most aggressive configuration that is possible and follows the assumption that the attacker has already gained knowledge about vulnerabilities. In two out of three tests, the attack was not able to find the target host fast enough. Anyhow, if the attack found and identified the target, it was able to penetrate the host successfully. Therefore, the host scan was further optimized to run in several different threads.

Worms: As of now we have looked at exploitation of singe hosts within enclosed segments of MTD-protected networks. In a production network, worms that spread from one segment to another pose a realistic threat that we also need to evaluate against [34]. As it is hard to find and deploy historical worms and effectively measure their propagation, a new proof-of-concept worm, named Networm, has been created and open-sourced for benchmark purposes of network-level MTD approaches [28]. Networm is also able to provide information about its propagation across the network.

Also, it will not use any software-dependent vulnerabilities. It will launch a brute-force attack on the SSH endpoint login data. The new worm can be classified according to the taxonomy created by Weaver et. al. [34] using i) Target Discovery, ii) Propagation Carriers and Distribution Mechanics, iii) Activation, iv) Payload, and v) Ecology.

Target Discovery: Nmap is implemented in insane mode as this is the most aggressive scanning mode available. The subnets for host discovery are chosen based on the configuration of the network interfaces and the routes. Thus, all locally connected subnets and targets can be scanned. It is defined which subnets are not allowed to be scanned explicitly to limit the possible worm propagation to the test environment. All detected subnets get split into several subnets and are randomly scanned, to avoid simultaneous scans by multiple worms. Propagation Carriers and Distribution Mechanics: The worm can be classified as self-carried because it is able to propagate itself to another host. Once the SSH credentials are correctly guessed, it is possible to establish a connection to the server. If the host has not already been infected, all necessary files will be transferred using the SSH File Transfer Protocol (SFTP) and will afterwards execute the worm on the newly infected host. After that, no connection from the attacking to the newly infected host is required anymore. Activation: Since the infecting host launches the worm, it can be classified as a self-activation worm. Payload: The worm does not need to execute any further processes besides the propagation to other hosts. The payload is classified as non-functional. Ecology: Given that every host needs to have simple guessable root login credentials, the environment is classified as monoculture.

To be able to evaluate the behaviour of the worm, it transfers a victim log containing the IP addresses and the current timestamp of each infection to the command and control server and every newly infected host. The command and control server is not part of the test network and therefore not shown in the simplified representation of the worm propagation. Therefore, it is possible to read the worm propagation routes directly from a folder on the server.

The test network has been expanded for the worm propagation experiment. The network, depicted on Figure 13 and 14, has been divided into three parts: green, blue, and red. The worm will be deployed to one machine in the red non-MT area of the network.

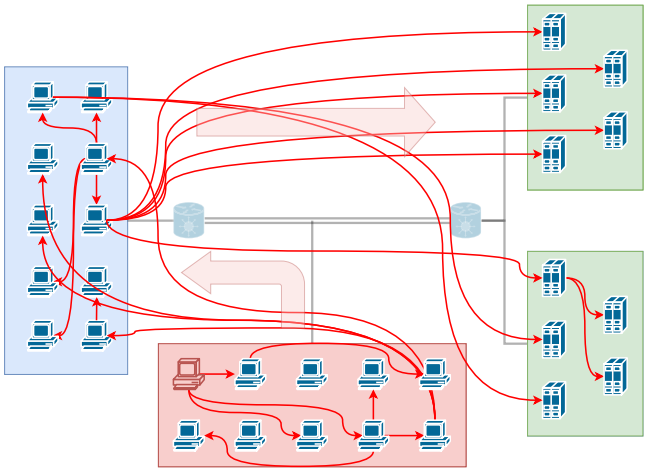


Figure 13: Worm Attacks - Process of Spreading Networm with PH

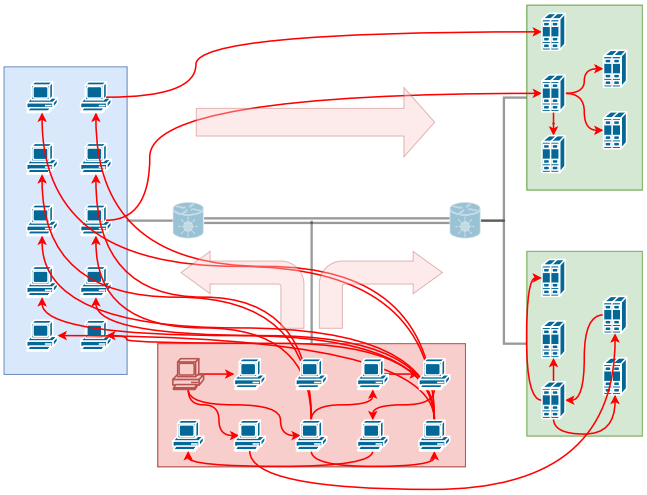


Figure 14: Worm Attacks - Process of Spreading Networm with NAS

This could be, for example, an internal company network. This network is for employees without security clearance. While the blue segment for clients with security clearance. They have a valid DNS configuration and know the domain names of NAS-protected servers. Additionally, they are able to communicate with PH via the client-side MT-Port-Gateway. Every part of the network contains ten hosts. Finally, the green server network is split into two subnets with separate MT-Gateways.

Port Hopping: Similar to Section 4.1, the first test was carried out activating the PH protection. Thus, it is not possible for hosts in the red segment to contact servers using standard ports. The results depicted on Figure 13 show that the worm was able to spread in the red section, which proves that the worm is working properly. Moreover, one host was able to detect and infect a host in the blue section. Next, one host of the blue network found the upper subnet of the green network infecting all their hosts. This shows that the worms can only take the gateway-to-gateway route to infect hosts. In other words, it was not possible to infect the green server network directly from the green network, effectively adding an additional layer of defense.

Network Address Shuffling: Unlike with PH, during the tests with NAS, it was possible to spread the worm from each network to the next. Initially, the worms in the red area infect each other. Next, two hosts of the blue network were able to infect the green server network. Each host was only able to infect one other host. A host in the red segment has infected the lower green network. Again only one host of the green network has been infected directly. Thus, the experiment shows that it was not possible to infect more than one host at once, given the high scanning speed and an easy challenge (brute-forcing only a few passwords). Even shorter HFM shuffling cycles (e.g., 10 s) did not alter the behaviour.

5 CONCLUSION

In this paper, we proposed OpenMTD, a framework for efficient MTD evaluation. In detail, we presented the following contributions: (i) Development of OpenMTD, a modular MTD evaluation platform that allows for ease of reproducibility which consists of a virtualized testbed and a software framework; (ii) Implementation of PH and NAS in a hybrid approach that leaves it up to the operator's discretion which features to use; (iii) Implementation of additional modules, e.g. connection tracking and honeypots, to extend current approaches. All of those parts have been made publicly accessible to enable coherent evaluations of other approaches in the MTD community. Moreover, (iv) we have evaluated PH, NAS, and our hybrid approach within OpenMTD against each other. On the one hand, it turned out that a heavy performance impact has to be tolerated once any of those methods is deployed. On the other hand, the level of security rapidly increases within an MTD protected network as not a single out-of-the-box tool, but only our (v) self-written Python worm, which performed very aggressive scanning and immediate exploitation afterwards, was able to penetrate the additional layer of defense. At the current development stage, we therefore suggest to deploy any of the evaluated MTD solutions only in networks that host high-value targets. Cost of deployment is high, but the added security benefit is considerable. We hope that OpenMTD will be used by fellow researchers in the field to evaluate their ideas and provide some modules in this research as a starting point.

As future work, we plan to pushing in-simulation advancements and implement MTD approaches on real hardware in our IoT Lab. We expect to reveal how much overhead is actually introduced by using simulation environments. Implementations on real hardware call for significant additional work, but will bring the operator community even more confidence in MTD approaches once shown that not only networks within simulations, but also on real hardware can benefit from this new branch of research.

ACKNOWLEDGEMENTS

The authors would like to thank the Chair for Communication Systems and Network Security as well as the Research Institute CODE for their comments and improvements. Research supported, in parts, by EU H2020 Project CONCORDIA GA 830927.

REFERENCES

[1] Ehab Al-Shaer, Qi Duan, and Jafar Haadi Jafarian. 2013. Random Host Mutation for Moving Target Defense. In *Security and Privacy in Communication Networks*, Angelos D. Keromytis and Roberto Di Pietro (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 310–327.

[2] S. Antonatos, P. Akritidis, E. P. Markatos, and K. G. Anagnostakis. 2005. Defending against Hitlist Worms Using Network Address Space Randomization. In *Proceedings of the 2005 ACM Workshop on Rapid Malcode*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1103626.1103633

[3] Gal Badishi, Amir Herzberg, and Idit Keidar. 2005. Keeping Denial-of-Service Attackers in the Dark. In *Distributed Computing*, Pierre Fraigniaud (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 18–32.

[4] Cisco Systems, Inc. 2018. Cisco Global Cloud Index: Forecast and Methodology, 2016–2021 White Paper. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html Last accessed 29 January 2020.

[5] Warren Connell, Massimiliano Albanese, and Sridhar Venkatesan. 2017. A Framework for Moving Target Defense Quantification. In *ICT Systems Security and Privacy Protection*, Sabrina De Capitani di Vimercati and Fabio Martinelli (Eds.). Springer International Publishing, Cham, 124–138.

[6] curl Team. [n.d.]. curl - the man page. https://curl.haxx.se/docs/manpage.html Last accessed 4 March 2020.

[7] Matthew Dunlop, Stephen Groat, William Urbanski, Randy Marchany, and Joseph Tront. 2011. MT6D: A Moving Target IPv6 Defense. In *2011 - MILCOM 2011 Military Communications Conference*. 1321–1326. https://doi.org/10.1109/MILCOM.2011.6127486

[8] Daniel Fraunholz, Daniel Krohmer, Simon Duque Anton, and Hans Dieter Schotten. 2018. Catch Me If You Can: Dynamic Concealment of Network Entities. In *Proceedings of the 5th ACM Workshop on Moving Target Defense* (Toronto, Canada) *(MTD '18)*. Association for Computing Machinery, New York, NY, USA, 31–39. https://doi.org/10.1145/3268966.3268970

[9] Daniel Fraunholz, Daniel Reti, Simon Duque Anton, and Hans Dieter Schotten. 2018. Cloxy: A Context-Aware Deception-as-a-Service Reverse Proxy for Web Services. In *Proceedings of the 5th ACM Workshop on Moving Target Defense* (Toronto, Canada) *(MTD '18)*. Association for Computing Machinery, New York, NY, USA, 40–47. https://doi.org/10.1145/3268966.3268973

[10] Johann George. [n.d.]. qperf - Linux man page. https://linux.die.net/man/1/qperf Last accessed 4 March 2020.

[11] GNS3. 2020. GNS3 Documentation. https://docs.gns3.com/ Last accessed 16 February 2020.

[12] Kousaburou Hari and Tadashi Dohi. 2010. Sensitivity Analysis of Random Port Hopping. In *2010 7th International Conference on Ubiquitous Intelligence Computing and 7th International Conference on Autonomic Trusted Computing*. 316–321. https://doi.org/10.1109/UIC-ATC.2010.69

[13] Internet Systems Consortium, Inc. [n.d.]. dig - Linux man page. https://linux.die.net/man/1/dig Last accessed 5 March 2020.

[14] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. 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* (Helsinki, Finland) *(HotSDN '12)*. Association for Computing Machinery, New York, NY, USA, 127–132. https://doi.org/10.1145/2342441.2342467

[15] Jafar Haadi H. Jafarian, Ehab Al-Shaer, and Qi Duan. 2014. Spatio-Temporal Address Mutation for Proactive Cyber Agility against Sophisticated Attackers. In *Proceedings of the First ACM Workshop on Moving Target Defense* (Scottsdale, Arizona, USA) *(MTD '14)*. Association for Computing Machinery, New York, NY, USA, 69–78. https://doi.org/10.1145/2663474.2663483

[16] Hyo jin Lee, Myung sup Kim, James W. Hong, and Gil haeng Lee. 2002. QoS Parameters to Network Performance Metrics Mapping for SLA Monitoring. http://mail.apnoms.org/knom/knom-review/v5n2/4.pdf

[17] Dorene Kewley, Russ Fink, John Lowry, and Mike Dean. 2001. Dynamic approaches to thwart adversary intelligence gathering. In *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, Vol. 1. 176–185 vol.1. https://doi.org/10.1109/DISCEX.2001.932214

[18] Henry C. J. Lee and Vrizlynn L. L. Thing. 2004. Port hopping for resilient networks. In *IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004*, Vol. 5. 3291–3295 Vol. 5. https://doi.org/10.1109/VETECF.2004.1404672

[19] Lockheed Martin Corporation. 2015. Cisco Global Cloud Index: Forecast and Methodology, 2016–2021 White Paper. https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/Gaining_the_Advantage_Cyber_Kill_Chain.pdf Last accessed 13 February 2020.

[20] Yue-Bin Luo, Baosheng Wang, and Gui-Lin Cai. 2015. Analysis of Port Hopping for Proactive Cyber Defense. In *International Journal of Security and Its Applications*, Vol. 9. 123–134. https://doi.org/10.14257/ijsia.2015.9.2.12

[21] Gordon Lyon. [n.d.]. Nmap Referece Guide. https://nmap.org/book/man.html Last accessed 7 March 2020.

[22] David Mills. 2008. Network Time Synchronization Research Project. https://www.eecis.udel.edu/~mills/ntp.html Last accessed 25 February 2020.

[23] Jeffrey Mogul and Jon Postel. 1985. *Internet Standard Subnetting Procedure*. STD 5. RFC Editor. http://www.rfc-editor.org/rfc/rfc950.txt http://www.rfc-editor.org/rfc/rfc950.txt.

[24] R. Mohtasin, P. W. C. Prasad, A. Alsadoon, G. Zajko, A. Elchouemi, and A. K. Singh. 2016. Development of a virtualized networking lab using GNS3 and VMware workstation. In *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. 603–609. https://doi.org/10.1109/WiSPNET.2016.7566205

[25] Jon Postel. 1980. *User Datagram Protocol*. STD 6. RFC Editor. http://www.rfc-editor.org/rfc/rfc768.txt http://www.rfc-editor.org/rfc/rfc768.txt.

[26] Jon Postel. 1981. *Transmission Control Protocol*. STD 7. RFC Editor. http://www.rfc-editor.org/rfc/rfc793.txt http://www.rfc-editor.org/rfc/rfc793.txt.

[27] r00t 3xp10it. [n.d.]. mass_exploiter. https://gist.github.com/r00t-3xp10it/7278942915a0514cecd73fd94a070b42 Last accessed 7 March 2020.

[28] Poschinger Richard. 2020. NetWorm, a benchmarking worm for MTD analysis. https://github.com/rposchinger/networm_py3

[29] Poschinger Richard. 2020. OpenMTD: A framework for efficient MTD evaluation. https://github.com/rposchinger/OpenMTD

[30] Boris Rogier. 2016. Measuring Network Performance: Links Between Latency, Throughput and Packet Loss. https://accedian.com/enterprises/blog/measuring-network-performance-latency-throughput-packet-loss/ Last accessed 5 March 2020.

[31] H. Singh, W. Beebee, and E. Nordmark. 2010. *IPv6 Subnet Model: The Relationship between Links and Subnet Prefixes*. RFC 5942. RFC Editor. http://www.rfc-editor.org/rfc/rfc5942.txt http://www.rfc-editor.org/rfc/rfc5942.txt.

[32] The Apache Software Foundation. [n.d.]. ab - Apache HTTP server benchmarking tool. https://httpd.apache.org/docs/2.4/programs/ab.html Last accessed 5 March 2020.

[33] Gernot Vormayr. 2019. fnfqueue. https://pypi.org/project/fnfqueue/ Last accessed 23 February 2020.

[34] Nicholas Weaver, Vern Paxson, Stuart Staniford, and Robert Cunningham. 2003. A Taxonomy of Computer Worms. In *Proceedings of the 2003 ACM Workshop on Rapid Malcode* (Washington, DC, USA) *(WORM '03)*. Association for Computing Machinery, New York, NY, USA, 11–18. https://doi.org/10.1145/948187.948190

[35] Justin Yackoski, Peng Xie, Harry Bullen, Jason Li, and Kun Sun. 2011. A Self-shielding Dynamic Network Architecture. In *2011 - MILCOM 2011 Military Communications Conference*. 1381–1386. https://doi.org/10.1109/MILCOM.2011.6127498