# Attack simulation for a realistic evaluation and comparison of network security techniques

Alexander Bajic and Georg T. Becker

Digital Society Institute, ESMT Berlin, Germany *
firstname.lastname@esmt.org

**Abstract.** New network security techniques and strategies, such as Moving Target Defense (MTD), with promising narratives and concepts emerge on a regular basis. From a practical point of view, some of the most essential questions in judging a new defense technique are: What kind of attacks — and under which conditions — can be prevented? How does it compare to the state-of-the-art? Are there scenarios in which this technique poses a risk? Answering these questions is often difficult and no common framework for evaluating new techniques exists today.
In this paper we present an early operational version of such a practical evaluation framework that is able to incorporate static and dynamic defenses alike. The main idea is to model realistic networks and attacks with a high level of detail, integrate different defenses into this model, and measure their contribution to security in a given scenario with the help of simulation. To show the validity of our approach we use a small but realistic enterprise network as a case study in which we incorporate different realizations of the MTD technique VM migration. The quantitative results of the simulation based on attacker revenue reveal that VM migration actually has a negative impact on security. Using the log files containing the individual attack steps of the simulation, a qualitative analysis is performed to understand the reason. This combination of quantitative and qualitative analysis options is one of the main benefits of using attack simulation as an evaluation tool.

**Keywords:** Moving target defense, attack simulation, attack graphs, network modeling

## 1 Introduction

How to secure networks and systems against malicious actors is an extremely important question in today's digitized world. It is not surprising that new approaches and techniques are being proposed on a regular basis. One of the current trends in network security is Moving Target Defense (MTD). The idea of MTD is

* This research is supported by Rheinmetall. To be published in the proceedings of the "Nordic Conference on Secure IT Systems" (NordSec 2018), Springer LNCS, Oslo, Norway, November 2018.

to not treat systems as static but dynamically change their appearance in ways that make reconnaissance and attacks considerably harder in practice. Several approaches for the network layer have been proposed. Maybe the most popular being network address space randomization both at the IP as well as at the MAC layer [1, 6, 15–17]. The increasing use of software defined networks (SDN) gives further rise to this development [17, 14]. Other examples of network-based MTD techniques are dynamic resource mapping [10] or the dynamic movement of anomaly detectors across a network [25].

However, measuring the effectiveness of these techniques is very difficult. A considerable amount of published work looks at individual techniques, mainly in a theoretical fashion. Still, the practical benefits of each technique are not always apparent and general evaluation techniques applicable to a broader range of defenses are yet to arrive. Furthermore, how the theoretical results map to specific use cases and scenarios is often not obvious, making it difficult for practitioners to evaluate a new security technique. Evaluation techniques that allow for an easy application on real-world scenarios while preserving a solid theoretical foundation could fill this gap.

In this paper we propose the utilization of attack simulation as an evaluation technique to approach this topic. The simulation is based on a detailed model of a realistic network as well as attack and defense actions. In contrast to previous modeling approaches, attack actions do not only comprise vulnerability exploits but also legitimate actions that have been observed in the real world and affect the attackers progress during the phases of an attack. The results of such simulation allow for the detailed inspection of interaction to unveil how attacks and defenses counteract each other in practice. Additionally, they show the extent to which a specific defense and its timing impact an attacker's success with regard to achieved goals and costs.

### 1.1 Main contribution

The main contribution can be summarized as follows:

- We propose attack simulation based on realistic vulnerabilities and attack steps as a method to evaluate dynamic defense techniques as advocated by MTD, as well as static countermeasures against network-based attacks. The level of detail exceeds that of previous work.
- We demonstrate the general applicability and usefulness of this approach using a case study consisting of a small enterprise network and different defense techniques (VM migration, VM resetting and IP shuffling). By providing results on measures such as attacker revenue and time spent, as well as detailed information of started and performed actions, our approach allows for both qualitative and quantitative analysis.
- Our evaluation of VM migration raises strong doubts if VM migration is a useful defense technique for corporate networks. Depending on the scenario, it can have a negative as well as positive impact on security. However, the positive impact was only an increase in attack time or decrease in the

attacker's success probability. In comparison, the negative impact was that new attack paths became available, resulting in the attacker achieving attack goals that were otherwise impossible to achieve in the given scenario.

## 2 Related work

In the MTD community various analysis techniques have been proposed specifically designed to analyze and/or compare dynamic network security techniques. Anderson *et al.* [3] present two mathematical models, one based on closed forms and another one based on Stochastic Petri Nets, to evaluate the effect of dynamic platforms as a defense on the success of attacks. Maleki *et al.* [18] utilize Markov models to investigate the effect of IP address randomization on attacker success. Connell *et al.* [5] focus on the trade-off between costs and gain by modeling both the costs of a defense as well as the security gain of the defense to find an optimum. As a case study they use VM migration with VM resetting, the same defense we employ in our case study.

These approaches are mathematically sound, but assume a very simplified attacker and defender model by reducing the investigated technique to the probabilistic effect of changing a single parameter. In how far these results are applicable to real world attacks remains unclear. Besides probabilistic models, also game-theoretic approaches have been suggested to analyze MTD techniques. Examples are the works of Prakash and Wellman [21], and Vadlamudi *et al.* [24]. Zhuang *et al.* [29] presented an approach to analyze the effectiveness of MTD techniques based on network graphs in which edges represent the compromisation of adjacent nodes. The model they suggest evaluates the likelihood of successful attacks with and without MTD techniques. Zaffarano *et al.* [28] present a framework to develop metrics for potentially relevant measures. These metrics are derived from raw data that have been gathered in different virtual environments with different attacker and defender objectives. Though neither reproducible nor extensible, users of this framework can investigate the effect of changes to a system through the sheer amount of data. A refined and more specific version of this framework is presented by Taylor *et al.* [23]. The framework proposed by Connell *et al.* [4] puts strong emphasis on the formal description of the mathematical model that allows for quantitative comparability. Yet, their approach is static so that not all MTD techniques can be described.

Zhuang *et al.* [30] suggest to utilize simulation on the basis of conservative attack graphs. They analyzed both VM-Shuffling and IP-Shuffling with help of the by now discontinued Nessi2 platform [22]. Though Zhuang *et al.* do not elaborate on the possibility to incorporate different defenses for the sake of comparative evaluation, they show that simulation on the basis of state descriptions is viable. Hong and Kim [10] proposed to analyze MTD techniques with help of their hierarchical attack representation model (HARM) that is based on attack trees and graphs that are arranged on different layers. They do so for VM migration, OS diversification, and VM resetting to demonstrate their effectiveness. However, their assessment does not consider continuous movement. They only

investigate whether or not a threat level can be reduced with the help of a configuration change that is induced by one of the aforementioned techniques. This way they turn the general question on when and how to move into an optimization problem for a known threat, thus movement will ultimately stop as soon as no further optimization is possible.

There exists a considerable amount of work on how to model attack steps to create attack trees and graphs as well as their subsequent analysis, be it static or based on simulation. Traditionally, attack graphs and trees are used to evaluate the security of networks or systems with regard to a specific goal and not to compare different defense techniques. Yet, by comparing an attack tree with variations of itself that consider the presence of different defensive techniques, one might derive a technique's impact on reaching the defined goal. Our solution is heavily influenced by these works.

Complete frameworks that describe networks with the help of modeling languages, automate tree/graph generation, and also perform quantitative evaluation are, for example, the TVA tool [12], MulVal [20], CAULDRON [11], CySeMoL [9], and P$^2$CySeMoL [8]. However, many approaches (e.g. MulVal and Cauldron) aim for automated modeling with help of network scanners and automatically generated exploit rules based on data from CVSS databases. While this is suitable to automatically analyze large networks, it does not offer the required level of detail for simulating complex interaction or attacker knowledge. Additionally, most frameworks do not consider the possibility of intermediate state changes caused by dynamic defense techniques and the effect this has on the corresponding attack graph. In the presence of an active defender, such attack graphs require repeated renewal. Therefore, frameworks that rely on only one intially generated attack graph are not well suited to analyze the effects of dynamic defenses. A modeling approach which focuses on processes rather than states is used in pwnPr3d [13, 26]. These processes are directly translated into attack steps. While it is capable of modeling dependencies of exploits and legitimate actions in high detail, it appeared not to be trivial to do so for interaction of attacker and defender.

## 3 Attack Simulation as an MTD evaluation tool

The primary goal is to get a realistic assessment of how various defenses perform in different scenarios. As Evans *et al.* [7] have pointed out, utility of a specific defensive technique is not universal but highly dependent on the context it is used in. Additionally, such an investigation on performance should not be limited to static defenses but incorporate a dynamic defender as is advocated by Moving Target Defense. Simulation appears to be a suitable approach as it is capable of incorporating numerous actors and can be applied to arbitrarily detailed scenarios. This allows for measuring the attacker's success rate and revenue in the presence of different defense techniques and, in turn, helps to determine which of these techniques is most useful in which network scenario.

### 3.1 Modeling networks, exploits and defenses

In the presence of the various approaches to modeling and evaluation, part of which have been shortly introduced in Section 2, we ultimately decided to employ deductive reasoning on the basis of coherent state descriptions with the help of Prolog, similar to MulVal. But unlike MulVal, we employ a more elaborate model of attack steps and state descriptions. Since detailed models require considerable effort when describing systems and actions, we first defined a high-level language. This language is human-readable and can automatically be translated to Prolog facts and rules. This has proven to be much more efficient than defining the system directly in Prolog.

A crucial aspect of realistic modeling is the handling of attacker knowledge, especially with regard to multi-stage attacks such as APTs (Advanced Persistent Threats), where lateral movement through a network plays an important role. Such movement is not only dependent on successful attacks but might equally be enabled by previously gained information and the use of legitimate functions. In realistic attacks, this is as important as exploits.

Key features of our modeling approach that achieve a higher level of detail compared to HARM [10] or attack graph tools such MulVal [20] are:

- Modeling of attacker knowledge, i.e. IP-addresses, DNS names, usernames, passwords, and other useful data (e.g. files representing attack goals).
- Modeling of legitimate functions such as database queries, remote shell, DNS lookups and ARP cache queries.
- Each exploit is modeled manually according to CVSS or metasploit descriptions and not simply based on the CVSS score and binary patch statuses.
- The results of exploits and attacker actions are freely programmable. This way exploits are not restricted to grant remote code execution (RCE) privileges or reveal credentials but more complex functionality such as return all data in RAM (e.g. for Meltdown) is feasible as well.

### 3.2 Attack simulation

The attack simulation itself is performed in a round-based fashion. Each attacker and defender action takes a certain amount of rounds to execute and has a success probability. In each round the defender acts first, followed by the attacker. To be more precise, the simulation algorithm is as follows:

**For $n$ rounds repeat:**

1. **Defender Actions**
   (a) For each defender action that is due in the current round do the following: If it is a probabilistic action, use a dice roll to determine if the action is successful or not. If so, perform the action and modify the state of the system accordingly. Finally, remove the action, no matter if successful or not, from the list of ongoing actions.

(b) Check if new defense actions are available (by checking if an action is feasible to execute as well as if it is in-line with the defense strategy). For each new defense action determine the finishing time and check if it has probabilistic parameters such as the direction of the shuffle. If so, use a dice roll to determine the parameters. Then add them to the list of ongoing defense actions.

2. **Attacker action**
   (a) For each action in the list of ongoing attack actions, check if it is still feasible in the current state. If not, because of a previous defense action, for example, remove the action from the list of ongoing attacker actions.
   (b) For each attacker action that is due in the current round do the following: If it is a probabilistic action, use a dice roll to determine if it is successful or not. If so, perform the action and modify the state of the system accordingly. Finally, remove the action, no matter if successful or not, from the list of ongoing actions.
   (c) Check if new attacker actions have become available that are not already in the list of ongoing actions. If so, calculate their finishing time and add them to the list of ongoing attack actions.

In our simulation approach we assume that the attacker can perform all available actions in parallel. However, once an action has been started, the same action cannot be initiated with the same parameters again as long as it is in the list of ongoing attacker actions. To illustrate this, an attacker can start a phishing attack against five different targets in a single round. However, once he has started a phishing attack against a target, the attacker has to wait until this phishing attack was either successful, failed or was defended before launching another phishing attack against the same target. But if the attacker learns of a new target, he is free to start a phishing attack against this new target any time.

For each round the simulation engine stores the generated revenue and relates the accumulated amount to the number of rounds that it took the attacker to reach it. Hence, the simulation engine does not report costs on a per action basis but counts the overall time till compromise, similar to the method used by P$^2$CySeMoL [8] and pwnPr3d [13]. The alternative would be to limit the number of actions an attacker can execute in parallel and assign costs to each action. However, this would require an intelligent attacker with a strategy, as it would be crucial for the attacker to choose the correct action at the correct time. Furthermore, many attacks can be automated so that — after the initial development — actually executing them might be a matter of starting a script and waiting. We, therefore, opted for a simulation in which each attack option is initiated whenever it becomes available. This leads to a fairer comparison since the results do not depend on how well the attacker AI has been tuned to a defense technique. Note, however, that analyzing some countermeasures such as honeypots requires the modeling of a smart attack. Modeling smart attackers for such cases is interesting future research.

## 4 Case Study

As a case study for our simulation-based evaluation approach we use VM shuffling, VM resetting, IP shuffling, and combinations thereof as defenses. In the absence of established benchmark networks we modeled our own reference network, which is based on a typical layout for small enterprises and employs commonly used applications and services.

### 4.1 Defense techniques

One of the MTD techniques we employ is frequently refered to as VM migration or shuffling that is basically the (random) relocation of VMs across various physical hosts. The idea has been addressed in several articles dealing with MTD and network defense in general [19, 2, 27]. It has also been used by Hong and Kim [10] as well as Connell *et al.* [5] in their MTD assessment methodologies. Hong and Kim assume the entire virtual node to be moved from one physical host to another using live migration. That means, the VM is moved without loosing its current state. We denote this defense technique as *live migration* in our experiments. In their case study, consisting of three hosts and seven VMs, this live migration changes the connectivity of the VMs which impacts the exploits that can be carried out by the attacker.

**Table 1.** The list of defenses used in our case study. In addition to these defenses, the scenario is also tested without any defenses.

| Name | Description | Impact |
|---|---|---|
| Live migration | The VM is migrated from one physical host to another without loosing its state. | Moving a VM changes the physical connectivity and hence the routing. |
| VM resetting | The VM is restarted from read-only memory, loosing all state information. | Any remote code execution privileges on the VM previously gained by the attacker are removed. |
| IP shuffling | A new IP address is assigned to the VM. | Knowledge of the IP address previously gained by the attacker is removed. |
| Cold migration | The VM is migrated to a different physical host, restarted there from read-only memory, and assigned a new IP. | This is the combination of live migration, VM resetting, and IP shuffling. |

Connell *et al.* [5] assume a different form of VM shuffling which we denote as *cold migration*. In their scenario, several VMs for the same applications can exist in parallel and the shuffle operation starts a new VM "from scratch" on a different host. New requests are then directed to this new VM and the old VM is shut down as soon as all old request are finished. The fact that VMs are shut

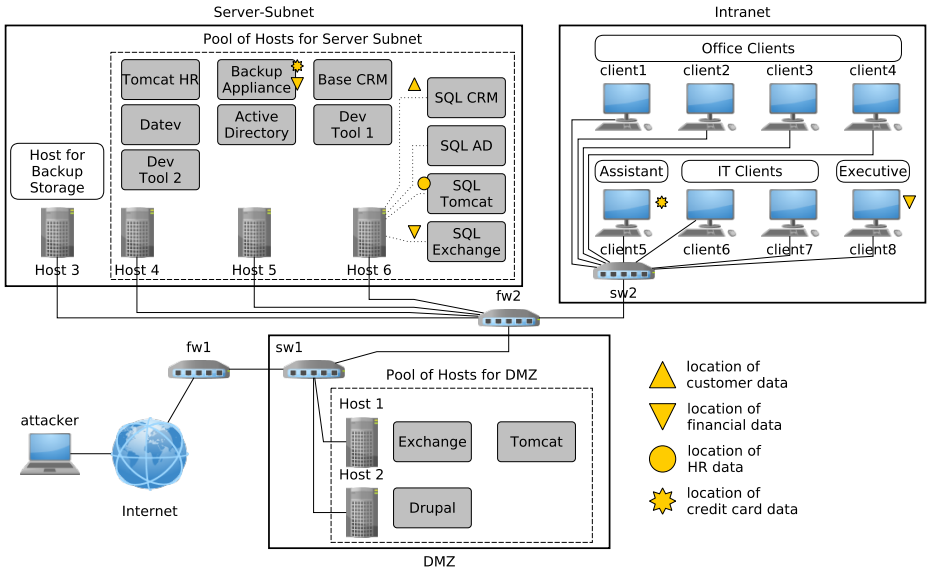


**Fig. 1.** The network used in our case study, representing a fairly typical small enterprise setup.

down and restarted from read-only memory ensures that attackers do not gain persistence on these servers. Furthermore, they assume some form of IP shuffling since each new VM gets a new IP via DHCP.

Note that this form of VM shuffling is much more difficult to realize in practice. It is only suitable for applications that do not need to persist data locally. In our analysis we ignore this aspect and will assume that each VM that can be shuffled using *live migration* can also be shuffled using *cold migration* to be able to compare the security of both approaches.

### 4.2 Network layout and software landscape

Figure 1 shows the network setup for the envisioned small enterprise network. The network is separated into a DMZ with servers accessible to the Internet, an intranet with clients, and a server subnet. The communication between zones as well as between machines in the server subnet is subject to firewalling. Furthermore, no machine beyond the DMZ is directly reachable from the Internet.

In the DMZ we assume two Xen servers that form a pool of hypervisors for three VMs. These comprise a Microsoft Exchange server running on Windows Server, and two VMs running on Ubuntu Server. One for the company's Drupal-based website, and one for a Tomcat server that hosts applications such as time tracking that are accessible to employees from the intranet as well as from the Internet after log-in.

In the server subnet we assume four hosts, three of which form another pool of Xen servers to host VMs, and one Ubuntu Server machine serving as a storage system for backups. The VMs in this second pool comprise:

- A Windows-based Active Directory Server acting as the domain controller, providing authentication services and network file sharing.
- A server running Base CRM, a proprietary customer relationship management system, based on Ubuntu Server.
- A server for accounting applications such as Datev, based on Windows Server.
- Another Tomcat server that exclusively runs applications for the HR department, based on Ubuntu Server.
- A Veritas Netbackup server to centrally command and control the backup agents on the various backup clients, based on CentOS server.
- Two Ubuntu-based servers for DevOps purposes (e.g. Jenkins and Jira).
- Four SQL servers, two of which are based on Ubuntu Server (for the Tomcat HR and Base CRM) and the other two being based on Windows Server (for Active Directory and Exchange).

Finally, we assume the client computers to be located in the intranet, which is connected to the server subnet and the DMZ through the second firewall. All clients are based on Windows 10 and differ in the user groups that operate them. They are equipped with the MS Office suite and backup agents.

In our example network, we modeled eight different attack goals that can be achieved by an attacker. All these goals are based on the retrieval of information, yielding different amounts of revenue which add up to 100 points in total. Four of these information elements are classified as "customer data", two of which yield 15 points each, the other two 10 points each. Additionally, there are two "financial data" elements, yielding 15 points each, as well as credit card information and HR data for 10 points, each. All of this data can be accessed in different ways. One way to access customer data is through the Base CRM frontend, if respective credentials have been obtained from the various back-office clients. Another option is to directly query the SQL server where data is stored, given the fact that the attacker was able to obtain username and password. Yet another possibility is to compromise the operating system that the SQL server is running on and exfiltrate the database. The financial data can be accessed through the CEO's computer or through his/her e-mail account, again opening up different ways to acquire this information. HR data can be obtained through compromising either the Tomcat server in the server subnet or the respective SQL server where data is stored. Finally, credit card information can be retrieved through access to the assistant's computer or its backup.

The fact that our sample network utilizes Xen hypervisors to host VMs for different purposes allows us to incorporate VM migration in our scenarios. From a practical point of view it does not make sense to shuffle VMs from the DMZ with those from the server subnet. Hence, we assume that VM shuffling is only used to move VMs across hosts that belong to the same pool. To simulate the

changed physical connectivity mentioned by Hong and Kim we chose to directly attach the hosts from the server subnet to the free ports of the routing firewall $FW2$. By default, most hypervisors use a virtual switch that the hosted VMs are attached to. We, therefore, assume that VMs located on the same host are connected by the virtual switch of the hypervisor and can communicate with each other regardless of the firewall setting in $FW2$. To summarize, the firewall $FW2$ limits the communication between VMs located on different hosts but the communication between VMs on the same host is not restricted.

It should be noted that the four VMs that serve as SQL servers for the different applications are never migrated but strictly allocated to host 6. This is due to the fact that the migration of VMs that contain large databases poses additional challenges in order to maintain availability and consistency. Additionally, VM resetting conflicts with the database's primary purpose to persist data.

### 4.3 Vulnerabilities and attack steps

Choosing realistic vulnerabilities and exploits, as well as legitimate actions that contribute to the attacker's progress is crucial for a fair and realistic evaluation. Hence, the question arises how to choose vulnerabilities, functions and exploits. Our approach was as follows: For the presented sample network we chose specific and commonly used software and searched the CVSS database and metasploit database for related entries since 2016. For each CVSS entry with a high score we manually checked if the vulnerability is likely to be applicable in our scenario. In particular, we chose to implement exploits that resulted either in remote code execution, privilege escalation or the retrieval of information (e.g credentials, RAM content etc.). We then manually modeled an exploit for the respective vulnerability with a high level of detail regarding its requirements and effects. Similarly, we manually modeled realistic legitimate functions of the assumed applications and systems which could also give the attacker execution rights or valuable information. Examples of such legitimate functions are remote shell for operating systems to gain remote code execution privileges, ARP-cache lookups to retrieve IP addresses, or SQL queries to retrieve information from databases.

One important aspect is defining the duration of an exploit as well as the attack success probability. Although the CVSS entries include parameters that are related to an exploit's duration and likelihood of success (e.g. attack complexity, exploit code maturity etc.), specific figures for these measures cannot be derived from a given score. We, therefore, manually determined values for duration and success rate based on a vulnerability's description, the underlying mechanism, and the availability of exploit code (e.g. in metasploit), noting that these values could potentially be optimized with data obtained from real world attacks.

In total we modeled 16 exploits as well as 10 legitimate functions an attacker can call, resulting in 26 executable functions from an attacker's perspective. Table 2 shows a few example functions to provide an impression of the level of detail used in our simulation. A table listing all functions with their requirements and effects can be found in the appendix.

**Table 2.** Example of attacker actions that were used in simulations that reached a revenue level of 100 points in the experiment depicted in Figure 3(c).

| |
|---|
| **Name**: phishingDocRCE (CVE-2016-0099) [exploit, metasploit exists]<br>**Result**: Attacker.remoteCodeExe+=App; **Time and probability**: 200 / 0.02,0.03, or 0.05 (depends on client)<br>**Requirements**: App=officeSuite & OS.family=windows & OS=App.parent & App.isPhishingVulnerable |
| **Name**: getCustomerData [legitimate function]<br>**Result**: Attacker.knows+=CRMUSER.data; **Time and probability**: 20 / 1<br>**Requirements**:App=baseCrm & CRMUSER=App.user & Attacker.knows=CRMUSER.password & Attacker.knows=CRMUSER.username & OS=App.parent & (Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) & Attacker.reachable(OS,Port=tcp) |
| **Name**: readData [helper function, next step after gaining RCE rights]<br>**Result**: Attacker.knows+=App.allData; **Time and probability**: 60 / 1<br>**Requirements**:App=Attacker.remoteCodeExe OR (OS=App.parent & OS=Attacker.remoteCodeExe) |
| **Name**: backupServerRCE (CVE-2016-7399) [exploit, metasploit exists]<br>**Result**: Attacker.remoteCodeExe+=OS & Attacker.knows+=App.backupedData;**Time and probability**: 33 / 1<br>**Requirements**: App=veritasBackupServer & OS=App.parent & OS.family=linux & App.hasCVE20167399 & (Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) & Attacker.reachable(OS,Port=tcp) |
| **Name**: sqlQuery [legitimate function]<br>**Result**: Attacker.knows+=USER.databaseData; **Time and probability**: 30 / 1<br>**Requirements**:App=sqlServer & USER=App.databaseUser & Attacker.knows=USER.password & Attacker.knows=USER.username & OS=App.parent & (Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) & Attacker.reachable(OS,Port=sqlport) |

## 5 Experimental results

We performed two independent experiments, one in which the attacker could utilize exploits based on vulnerabilities published in 2016 (4 exploits plus 10 legitimate functions) and one with exploits based on vulnerabilities from 2017/2018 (12 exploits and 10 legitimate functions). In both cases we tested the performance when using *no defense technique*, *live migration*, *cold migration*, *IP shuffling*, and *VM resetting*. Each simulation was started 100 times and consisted of 8000 rounds each. Furthermore, we defined three revenue thresholds of 40, 75 and 100 points respectively and measured how many simulations reached these thresholds for a given number of rounds. The results are depicted in Figure 2.

In the experiment where exploits from 2017 and 2018 were used, no significant difference between having *no defense technique*, *IP shuffling* or *live migration* can be observed. The attack was successful fairly quick and all simulations achieved the maximum revenue of 100 points. When *VM resetting* or *cold migration* were enabled, it took more rounds for the attacker to reach revenue levels of 75 or 100. Hence, one can say that they had a positive impact on security. *Cold migration* is the combination of *live migration*, *IP shuffling* and *VM resetting*. The fact that *cold migration* and *VM resetting* performed nearly identical indicates that the security gain primarily results from *VM resetting* and not from migrating (shuffling) VMs.

In the second experiment only four exploits and ten legitimate functions were available, resulting in fewer viable attack paths. In this case, all defenses performed similar for a revenue threshold of 40. However, revenue levels of 75 or 100

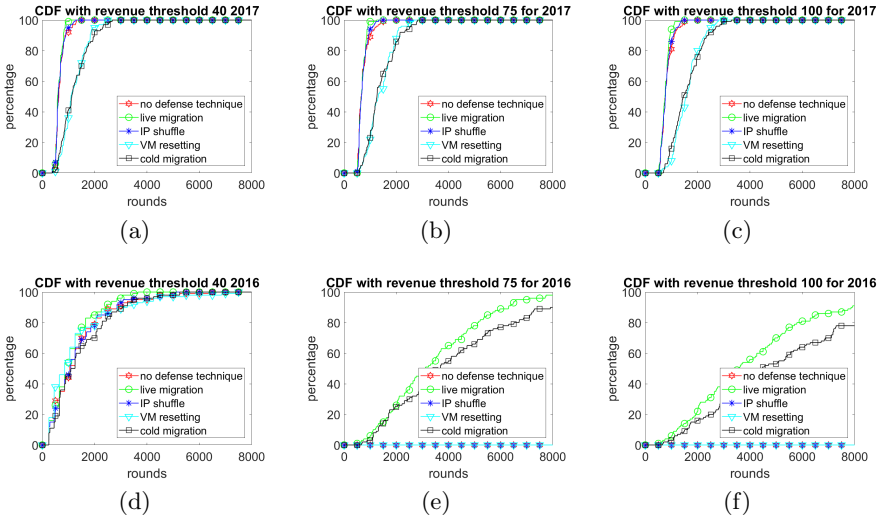


**Fig. 2.** Results of the attack simulation. Each defense was simulated 100 times for exploits based on 2017/2018 vulnerabilities (a-c) and 2016 vulnerabilities (d-f). Results are displayed with regard to reached threshold with the y-axis depicting the percentage of simulations that reached the respective success threshold for the given round (x-axis).

were only achieved when either *live migration* or *cold migration* was enabled. If *no defense technique*, *IP shuffling* or *VM resetting* was used, these revenue levels were never reached in any simulation run. The log data of the simulations reveal that there was only one possible attack path to achieve at least 75 points. The used attacker actions are listed in Table 2. The first step is that the attacker launches a successful phishing attack against one of the clients. The attacker can then use the remote code execution privileges as well as the stolen DNS names to launch an attack based on exploit "backupserverRCE" (CVE-2016-7399) on the backup server. These first attack steps are independent from any of the used defense techniques and generate more than 40 revenue points for the attacker. This is due to data directly found on the attacked client and backup server, as well as using the Base CRM server with stolen credentials of the client. Besides data that directly generates revenue, additional useful data is stored in the backup. In particular, it also contains configuration files of the "Base CRM" and "Tomcat HR" servers and the corresponding SQL credentials. These SQL credentials can then be used in the next attack step to retrieve customer data and HR data using regular SQL queries and database management commands. However, to perform these regular functions, the attacker needs to be able to communicate with the SQL servers on host 5 via the SQL port 3302. Both nodes that the attacker controls — the compromised client (phishing) as well as the backup server (backupserverRCE) — are not whitelisted to communicate on the SQL port. Since the backup server and the SQL server are located on different hosts at the beginning, the firewall blocks such communication attempts. Therefore,

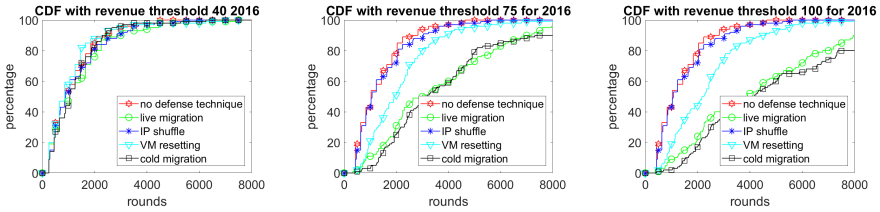


**Fig. 3.** The same analysis as in Figure 2 (d-f) but this time the starting position of the backup server was on host 6 (same as the SQL servers) instead of host 5.

for *no defense technique*, *IP shuffling* or *VM resetting* the attacker cannot call these SQL functions and, in consequence, never reaches revenue levels of 75 or above. However, the firewall cannot block communication between VMs on the same host as they are, by default, attached to the same virtual switch. The log data reveals that when *live migration* or *cold migration* is enabled, the backup server is shuffled to the same host as the SQL databases roughly one-third of the times. Hence, whenever the backup server was on the same host as the SQL database, the attacker could download the data using SQL queries until another shuffle operation migrated the backup server to a different host.

Please note that this scenario is exactly as discussed by Hong and Kim [10] to assess the effectiveness of live migration. The migration of VMs changes the physical connectivity and with it the attack paths. However, as our experiment shows, this can have a significant negative impact on security as the migration enables attack paths that would otherwise not exist.

### 5.1 Never trust a statistic you have not forged yourself

In our experiments depicted in Figure 2, migration had a negative impact on security. Only the removal of the attacker's RCE privileges (which is being done in *VM resetting* and *cold migration*) had a notably positive effect on security. However, resetting VMs only hindered the attacker and made attacks more difficult with regard to the required time (rounds) to a full compromise but could not completely fend off attacks. Of course, if resetting is done with a much higher frequency it is possible to get results in which all attacks are completely defended. However, such timings are not necessarily very realistic.

Indeed, one can also produce scenarios in which migration has a measurable positive effect. If we look at the experiment based on the 2016 exploits, the reason why the attack does not work if no migration is used is that the VMs of the backup server and the SQL servers are not on the same physical host. To generate positive results for migration, we therefore modified the starting position of VMs and moved the backup server to the same host as the SQL servers. Figure 3 shows the experimental results for this modified case for 2016. In this case, migration contributed to security. The reason is that with migration turned on, the backup server and the SQL server were on different hosts two-

thirds of the time, while for the other non-migrating defenses they were always on the same physical host. Hence, attacking was more difficult in that it took the attacker more rounds to exfiltrate the data. But please note that the attacker was still able to exfiltrate all data within 8000 rounds in 90% (*live migration*) and 80% (*cold migration*) of the simulations.

By tuning the scenario, one can heavily influence the results of the attack simulation. However, attack simulation not only outputs revenue data but also all attack steps performed by the attacker (i.e., log data). This data can be used to understand why a defense performed a certain way, which is exactly what we did to understand and describe the reason why migration performed so poorly for the 2016 scenario in Figure 2 (e,f). Hence, these log data allow for a qualitative analysis of the experiments which is useful to put the quantitative results into the correct context. We would like to point out that this combination of quantitative and qualitative analysis options is one of the great advantages compared to other evaluation techniques.

## 6 Conclusion and future work

In the course of our case study, we conducted experiments with moving target defenses based on VM shuffling/migration. We showed that while random VM migration can have a positive effect on security, it is more likely to have a negative impact on security. VM migration changes the physical connectivity and therefore influences attack steps. If the starting position is beneficial for the attacker, moving the VMs increases the attack time because the attacker has to wait until the VMs have shuffled back to a suitable position. However, if the starting position does not allow an attack, random VM migration will eventually shuffle the position such that an attack becomes feasible. That means, the potential positive impact is only an increase in attack time while the negative impact is that formerly impossible attacks now become feasible.

Hence, this case study shows that attack simulation based on realistic exploits, functions and network setups is indeed useful to analyze and compare defense techniques. One of the main benefits of this simulation approach is that with the same experiment both a quantitative analysis based on the attacker revenue as well as a qualitative analysis based on the log file of attack steps is possible. This combination ensures that one can put the quantitative results into the correct context.

To summarize, defense evaluation on the basis of attack simulation is not a technique that generates reliable results at the press of a button. Instead, one has to verify that the attack simulation models attacks and defenses with sufficient accuracy for the tested defense techniques. In addition, example networks and exploits must be selected to fit the intended application. Indeed, developing commonly accepted benchmarks consisting of a range of realistic networks and exploits would be a very useful future research direction for the network security community. But if one models the network and exploits with enough details and uses a suitable scenario, attack simulation is a very helpful tool to

evaluate and compare network security techniques. It is especially useful as a bridge between often quite theoretical research proposals and quantifiable and practically relevant results suitable for practitioners and decision makers.

## References


1. Al-Shaer, E., Duan, Q., Jafarian, J.H.: Random host mutation for moving target defense. In: International Conference on Security and Privacy in Communication Systems. pp. 310–327. Springer (2012)
2. Almohri, H.M.J., Watson, L.T., Evans, D.: Misery digraphs: Delaying intrusion attacks in obscure clouds. IEEE Transactions on Information Forensics and Security **13**(6), 1361–1375 (June 2018)
3. Anderson, N., Mitchell, R., Chen, I.R.: Parameterizing moving target defenses. In: 2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS). pp. 1–6 (Nov 2016)
4. Connell, W., Albanese, M., Venkatesan, S.: A framework for moving target defense quantification. In: IFIP International Conference on ICT Systems Security and Privacy Protection. pp. 124–138. Springer (2017)
5. Connell, W., Menascé, D.A., Albanese, M.: Performance modeling of moving target defenses. In: Proceedings of the 2017 Workshop on Moving Target Defense. pp. 53–63. MTD '17, ACM, New York, NY, USA (2017)
6. Dunlop, M., Groat, S., Urbanski, W., Marchany, R., Tront, J.: Mt6d: A moving target ipv6 defense. In: Military Communications Conference - MILCOM 2011. pp. 1321–1326 (Nov 2011)
7. Evans, D., Nguyen-Tuong, A., Knight, J.: Effectiveness of Moving Target Defenses, pp. 29–48. Springer (2011)
8. Holm, H., Shahzad, K., Buschle, M., Ekstedt, M.: $P^2$ CySeMoL: Predictive, probabilistic cyber security modeling language. IEEE Transactions on Dependable and Secure Computing **12**(6), 626–639 (Nov 2015)
9. Holm, H., Sommestad, T., Ekstedt, M., Nordström, L.: CySeMoL: A tool for cyber security analysis of enterprises. In: 22nd International Conference and Exhibition on Electricity Distribution (CIRED 2013). pp. 1–4 (June 2013)
10. Hong, J.B., Kim, D.S.: Assessing the effectiveness of moving target defenses using security models. IEEE Transactions on Dependable and Secure Computing **13**(2), 163–177 (March 2016)
11. Jajodia, S., Noel, S., Kalapa, P., Albanese, M., Williams, J.: Cauldron mission-centric cyber situational awareness with defense in depth. In: Military Communications Conference - MILCOM 2011. pp. 1339–1344 (2011)
12. Jajodia, S., Noel, S., O'Berry, B.: Topological Analysis of Network Attack Vulnerability, pp. 247–266. Springer US, Boston, MA (2005)
13. Johnson, P., Vernotte, A., Ekstedt, M., Lagerström, R.: pwnpr3d: an attack-graph-driven probabilistic threat-modeling approach. In: Availability, Reliability and Security (ARES), 2016 11th International Conference on. pp. 278–283. IEEE (2016)
14. Kampanakis, P., Perros, H., Beyene, T.: 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. pp. 1–6 (June 2014)
15. Kewley, D., Fink, R., Lowry, J., Dean, M.: Dynamic approaches to thwart adversary intelligence gathering. In: DARPA Information Survivability Conference amp; Exposition II, 2001. DISCEX '01. Proceedings. vol. 1, pp. 176–185 vol.1 (2001)

16. Li, J., Yackoski, J., Evancich, N.: Moving target defense: A journey from idea to product. In: Proceedings of the 2016 ACM Workshop on Moving Target Defense. pp. 69–79. MTD '16, ACM (2016)
17. MacFarland, D.C., Shue, C.A.: 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. pp. 37–41. MTD '15, ACM (2015)
18. Maleki, H., Valizadeh, S., Koch, W., Bestavros, A., van Dijk, M.: Markov modeling of moving target defense games. In: Proceedings of the 2016 ACM Workshop on Moving Target Defense. pp. 81–92. MTD '16, ACM (2016)
19. Neupane, R.L., Neely, T., Chettri, N., Vassell, M., Zhang, Y., Calyam, P., Durairajan, R.: Dolus: Cyber defense using pretense against ddos attacks in cloud platforms. In: Proceedings of the 19th International Conference on Distributed Computing and Networking. pp. 30:1–30:10. ICDCN '18, ACM (2018)
20. Ou, X., Govindavajhala, S., Appel, A.W.: Mulval: A logic-based network security analyzer. In: USENIX Security Symposium. pp. 8–8. Baltimore, MD (2005)
21. Prakash, A., Wellman, M.P.: Empirical game-theoretic analysis for moving target defense. In: Proceedings of the Second ACM Workshop on Moving Target Defense. pp. 57–65. MTD '15, ACM, New York, NY, USA (2015)
22. Schmidt, S., Bye, R., Chinnow, J., Bsufka, K., Camtepe, A., Albayrak, S.: Application-level simulation for network security. SIMULATION **86**(5-6), 311–330 (2010)
23. Taylor, J., Zaffarano, K., Koller, B., Bancroft, C., Syversen, J.: Automated effectiveness evaluation of moving target defenses: Metrics for missions and attacks. In: Proceedings of the 2016 ACM Workshop on Moving Target Defense. pp. 129–134. MTD '16, ACM, New York, NY, USA (2016)
24. Vadlamudi, S.G., Sengupta, S., Taguinod, M., Zhao, Z., Doupé, A., Ahn, G.J., Kambhampati, S.: Moving target defense for web applications using bayesian stackelberg games: (extended abstract). In: Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems. pp. 1377–1378. AAMAS '16 (2016)
25. Venkatesan, S., Albanese, M., Cybenko, G., Jajodia, S.: A moving target defense approach to disrupting stealthy botnets. In: Proceedings of the 2016 ACM Workshop on Moving Target Defense. pp. 37–46. MTD '16, ACM (2016)
26. Vernotte, A., Johnson, P., Ekstedt, M., Lagerstrm, R.: In-depth modeling of the unix operating system for architectural cyber security analysis. In: 2017 IEEE 21st International Enterprise Distributed Object Computing Workshop (EDOCW). pp. 127–136 (Oct 2017)
27. Wang, H., Li, F., Chen, S.: Towards cost-effective moving target defense against ddos and covert channel attacks. In: Proceedings of the 2016 ACM Workshop on Moving Target Defense. pp. 15–25. MTD '16, ACM, New York, NY, USA (2016)
28. Zaffarano, K., Taylor, J., Hamilton, S.: A quantitative framework for moving target defense effectiveness evaluation. In: Proceedings of the Second ACM Workshop on Moving Target Defense. pp. 3–10. MTD '15, ACM (2015)
29. Zhuang, R., DeLoach, S.A., Ou, X.: A model for analyzing the effect of moving target defenses on enterprise networks. In: Proceedings of the 9th Annual Cyber and Information Security Research Conference. pp. 73–76. CISR '14, ACM, New York, NY, USA (2014)
30. Zhuang, R., Zhang, S., DeLoach, S.A., Ou, X., Singhal, A.: Simulation-based approaches to studying effectiveness of moving-target network defense. In: National Symposium on Moving Target Research. NIST (2012)

# A Appendix

Table 4 lists the modeled exploits and Table 3 legitimate system functions that can be used by an attacker.

**Table 3.** Overview of attacker actions based on legitimate functions

| |
|---|
| **Name**: readData (helper function, next step after gaining RCE rights) |
| **Result**: Attacker.knows+=App.allData; **Time and probability**: 20 / 33 |
| **Requirements**:App=Attacker.remoteCodeExe OR (OS=App.parent & OS=Attacker.remoteCodeExe) |
| **Name**: pingscan |
| **Result**: Attacker.knows+=OS.ipaddress; **Time and probability**: 20 / 33 |
| **Requirements**: Attacker.reachable(OS,Port=ping) |
| **Name**: arpCache (gives attacker all IP-addresses of the subnet of a compromised system) |
| **Result**: Attacker.knows+=TARGET.ipaddress; **Time and probability**: 20 / 33 |
| **Requirements**:(App=Attacker.remoteCodeExe OR OS=Attacker.remoteCodeExe) & OS=App.parent & SUBNET=OS.belongsToSubnet & TARGET.belongsToSubnet=SUBNET |
| **Name**: configureActiveDirectoryClients |
| **Result**: Attacker.remoteCodeExe+=TARGET; **Time and probability**: 20 / 33 |
| **Requirements**: App=activeDirectory & Attacker.remoteCodeExe=App & TARGET=App.clients |
| **Name**: getCustomerData |
| **Result**: Attacker.knows+=CRMUSER.data; **Time and probability**: 20 / 33 |
| **Requirements**:App=baseCrm & CRMUSER=App.user & Attacker.knows=CRMUSER.password & Attacker.knows=CRMUSER.username & OS=App.parent & (Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) & Attacker.reachable(OS,Port=tcp) |
| **Name**: getMail |
| **Result**: Attacker.knows+=CRMUSER.data; **Time and probability**: 20 / 33 |
| **Requirements**:App=exchangeServer & EMAILUSER=App.user & Attacker.knows=EMAILUSER.password & Attacker.knows=EMAILUSER.username & OS=App.parent & (Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) & Attacker.reachable(OS,Port=tcp) |
| **Name**: remoteDatabaseManagement |
| **Result**: Attacker.knows+=App.allDatabaseData; **Time and probability**: 20 / 33 |
| **Requirements**:App=sqlServer & ADMIN=App.admin & Attacker.knows=ADMIN.password & Attacker.knows=ADMIN.username & OS=App.parent & (Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) & Attacker.reachable(OS,Port=SQLPORT) |
| **Name**: sqlQuery |
| **Result**: Attacker.knows+=USER.databaseData; **Time and probability**: 20 / 33 |
| **Requirements**:App=sqlServer & USER=App.databaseUser & Attacker.knows=USER.password & Attacker.knows=USER.username & OS=App.parent & (Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) & Attacker.reachable(OS,Port=sqlport) |
| **Name**: remoteShellLinux |
| **Result**: Attacker.remoteCodeExe+=OS; **Time and probability**: 20 / 33 |
| **Requirements**: OS.family=Linux & OS.remoteShellEnabled & ADMIN=OS.root & Attacker.knows=ADMIN.password & Attacker.knows=ADMIN.username & (Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) & Attacker.reachable(OS,Port=22) |
| **Name**: remoteShellWindows |
| **Result**: Attacker.remoteCodeExe+=OS; **Time and probability**: 20 / 33 |
| **Requirements**: OS.family=Windows & OS.remoteShellEnabled & ADMIN=OS.root & Attacker.knows=ADMIN.password & Attacker.knows=ADMIN.username & (Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) & Attacker.reachable(OS,Port=3389) |

**Table 4.** Overview of attacker actions based on exploits

| |
|---|
| **Name**: tomPrivEscalation (CVE-2016-9775, CVE2016-9774) |
| **Result**: Attacker.remoteCodeExe+=OS; **Time and probability**: 20 / 33 |
| **Requirements**: App=tomcat & App.hasTomPriv & Attacker.remoteCodeExe=App & OS=Linux |
| **Name**: privEscalationWindows (CVE-2016-0026) metasploit exists |
| **Result**: Attacker.remoteCodeExe+=OS; **Time and probability**: 20 / 33 |
| **Requirements**: (OS=Windows10 OR WindowsServer2016) & OS.hasWinPrivEscalation & Attacker.remoteCodeExe=App |
| **Name**: backupServerRCE (CVE-2016-7399) metasploit exists |
| **Result**: Attacker.remoteCodeExe+=OS; **Time and probability**: 20 / 33 |
| **Requirements**: App=veritasBackupServer & OS=App.parent & OS.family=linux & App.hasCVE20167399 & (Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) & Attacker.reachable(OS,Port=tcp) |
| **Name**: phishingDocRCE (CVE-2016-0099) metasploit exists |
| **Result**: Attacker.remoteCodeExe+=App; **Time and probability**: 20 / 33 |
| **Requirements**: App=officeSuite & OS.family=windows & OS=App.parent & App.isPhishingVulnerable |
| **Name**: tomHttpPutRCE (CVE-2017-12615, CVE-2017-12617) |
| **Result**: Attacker.remoteCodeExe+=App; **Time and probability**: 20 / 33 |
| **Requirements**: App=tomcat & App.hasHttpPutVulnerability & OS=App.parent &(Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) & Attacker.reachable(OS,Port=jmxport) |
| **Name**: jmxTomcatVulnerability (Blog 2017) |
| **Result**: Attacker.remoteCodeExe+=OS; **Time and probability**: 20 / 33 |
| **Requirements**: App=tomcat & App.hasJmxEnabled & OS=App.parent & ( Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName ) & (App.jmxNoAuth OR (Attacker.knowsUsername(App) & Attacker.knowsPassword(App)) & Attacker.reachable(OS,jmxport) |
| **Name**: privEscalationUbuntu (CVE-2017-0358) |
| **Result**: Attacker.remoteCodeExe+=OS; **Time and probability**: 20 / 33 |
| **Requirements**: OS=Ubuntu & OS.hasUbuntuPrivEscalation & OS=App.parent & Attacker.remoteCodeExe=App |
| **Name**: eternalBlueRCE (CVE-2017-0143 to 0148) metasploit exists |
| **Result**: Attacker.remoteCodeExe=OS; **Time and probability**: 20 / 33 |
| **Requirements**: OS.family=Windows & OS.hasEternalBlue & (Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) & Attacker.reachable(OS,Port=smb) |
| **Name**: redirectBackupToCloud (CVE-2017-6409) |
| **Result**: Attacker.knows+=App.backupedData; **Time and probability**: 20 / 33 |
| **Requirements**: App=veritasBackupServer & App.hasCloudVuln & (Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) & Attacker.reachable(OS,Port=tcp/5637) |
| **Name**: backupClientRCE (CVE-2017-8895) metasploit exists |
| **Result**: Attacker.remoteCodeExe=OS; **Time and probability**: 20 / 33 |
| **Requirements**: App=veritasBackupClient & Attacker.parent=OS & OS.family=windows & APP.hasSSLVuln & (Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) & Attacker.reachable(Os,Port=ssl) |
| **Name**: clientRCEoverServer (CVE-2017-6407) |
| **Result**: Attacker.remoteCodeExe=OS; **Time and probability**: 20 / 33 |
| **Requirements**: App=veritasBackupClient & App.hasRCEfromServer & SERVER=App.server & Attacker.remoteCodeExe=SERVER & OS=App.parent & Attacker.knows=OS.ipaddres & reachable(OS,Port=ssl) |
| **Name**: meltdown (CVE-2017-5715, 2017-5753) |
| **Result**: Attacker.knows+=Node.dataInRAM; **Time and probability**: |
| **Requirements**: NODE.type=Intel & OS.runsOn=NODE & OS.hasMeltdown & App.parent=OS & Attacker.remoteCodeExe=App |
| **Name**: drupalRCE (CVE-2017-5715, 2017-5753) metasploit exists |
| **Result**: Attacker.remoteCodeExe+=App ; **Time and probability**: |
| **Requirements**: App=drupal & App.hasRCEviaHttpGetVuln OS=App.parent & (Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) & Attacker.reachable(OS,Port=http) |
| **Name**: sendMailExchangeRCE (CVE-2018-8154) |
| **Result**: Attacker.remoteCodeExe+=App ; **Time and probability**: |
| **Requirements**: App=exchangeServer & OS=App.parent & OS.family=windows & USER=App.emailuser & Attacker.knows=USER.username & Attacker.knows=USER.password & (Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) |
| **Name**: exchangeDefenderRCE (CVE-2018-0986) |
| **Result**: Attacker.remoteCodeExe+=App; **Time and probability**: |
| **Requirements**: App=exchangeServer & OS=App.parent & (Attacker.knows=OS.ipaddress OR Attacker.knows=OS.dnsName) |