# Security Modeling and Analysis of Moving Target Defense in Software Defined Networks

Júlio Mendonça[1], Minjune Kim[2], Rafal Graczyk[1], Marcus Völp[1], Dan Dongseong Kim[2]
[1] *SnT, University of Luxembourg*, Luxembourg
[2]*School of IT and Electrical Engineering, University of Queensland*, Australia
julio.mendonca@uni.lu, mj.kim@uq.edu.au, {rafal.graczyk, marcus.voelp}@uni.lu, dan.kim@uq.edu.au

***Abstract*—The use of traditional defense mechanisms or intrusion detection systems presents a disadvantage for defenders against attackers since these mechanisms are essentially reactive. Moving target defense (MTD) has emerged as a proactive defense mechanism to reduce this disadvantage by randomly and continuously changing the attack surface of a system to confuse attackers. Although significant progress has been made recently in analyzing the security effectiveness of MTD mechanisms, critical gaps still exist, especially in maximizing security levels and estimating network reconfiguration speed for given attack power. In this paper, we propose a set of Petri Net models and use them to perform a comprehensive evaluation regarding key security metrics of Software-Defined Network (SDNs) based systems adopting a time-based MTD mechanism. We evaluate two use-case scenarios considering two different types of attacks to demonstrate the feasibility and applicability of our models. Our analyses showed that a time-based MTD mechanism could reduce the attackers' speed by at least 78% compared to a system without MTD. Also, in the best-case scenario, it can reduce the attack success probability by about ten times.**

***Index Terms*—Security, Moving Target Defense, Software Defined Networks, Petri nets**

This is a preprint version. The final version will appear on the PRDC'22 proceedings.

## I. Introduction

Traditional defense measures for dealing with malicious threats include installing anti-malware software, deploying firewalls, blocking network traffic and applications, and denying permissions to specific resources. Also, many intrusion detection systems (IDSs) have been developed to identify various kinds of attacks and trigger responsive actions. However, IDSs and traditional defense mechanisms are reactive. They act only after having identified malicious activities, which often is too late and therefore ineffective against agile and intelligent cybercriminals (e.g., attackers). Late defense actions increase the attackers' chances of launching effective and efficient cyberattacks since they have enough time to gather and exploit information [1].

Fortunately, Moving Target Defense (MTD) has been proposed as a proactive defense mechanism. It secures systems by introducing continuous change in the protected computer systems or their networks (e.g., by altering network settings, replacing the guest Operating System of a Virtual Machine (VM), or alternating versions of the hypervisor to execute a VM). In doing so, MTD aims to raise uncertainty in the attacker, requiring him to spend time adjusting the attack to the present configuration [2]. MTD techniques operate in an arbitrary combination of time- and event-triggered reconfigurations. A pure event-triggered MTD reconfigures the system only after a particular event occurs. Time-triggered MTDs act periodically at configurable rates. Furthermore, hybrid approaches [3] mix the previous options to act periodically but also in response to triggering events. See [4] for a recent survey of MTD techniques.

The philosophy behind MTD, in comparison to conventional cybersecurity techniques, is to assume systems will get compromised even if a lot of effort is spent on defending it. In consequence, MTD advocates adapting to threats by change. Classical cybersecurity aims for identifying new attack vectors and vulnerabilities, patching them as quickly as possible in order to raise more and more defenses. On the other hand, MTD concerns with ensuring sufficient diversity to avoid common mode failures while guaranteeing that the rate of change remains sufficiently high to outpace the adversary in compromising components. To that end, MTD techniques include shuffling network addresses (real and virtual) and ports, switching over VM and containers, and changing the hardware by selecting diverse subsets among a pool of spare nodes [4].

More recently, Sharma et al. [5], Alavizadeh et al. [6], and Alhozaimy and Menascé [7] have analyzed the effectiveness of MTD techniques, identifying two significant gaps: (1) finding the optimal time to trigger MTD operations, such that system security is maximized, is a largely unsolved task, particularly for time-triggered systems; and (2) attack speeds remain largely unknown and have proven difficult to estimate correctly. Therefore, in this paper, we set out to fill these gaps by proposing a set of Petri nets models to analyze the security aspects of systems, applying virtual IP (vIP) shuffling as a time-based MTD mechanism. Our proposed models consider crucial phases of cyber attacks as defined by the Cyber Kill Chain [8], such as *Reconnaissance* and *Exploitation*. The parameterized characteristics of the model also allow one to evaluate different behaviors of attackers, such as the number of scanning packets sent during an attack. The models also adopt the MTD technique proposed by Sharma et al. [5], namely Flexible Randomisation Virtual IP Multiplexing (FRVM), and extend it by relaxing the assumption that the attack rate and the rate to trigger the MTD operation should be equal.

Our results show substantial improvement in the security levels when adopting a time-based MTD mechanism. Among the analyzed scenarios, the security improvement can surpass 600% compared to a solution without MTD, but it comes with a high reconfiguration rate for the MTD operation. In the worst case, adopting an MTD mechanism increases the security results by at least 78% when applying a reconfiguration rate smaller than 600 seconds. Also, the two use cases show that the proposed models can be adopted and extended to evaluate different scenarios and types of attacks to analyze security trade-offs between different system settings. More precisely, our contributions are as follows:

- we proposed extendable, and easily usable, security models by means of a Petri net abstraction, capable of capturing different attackers' behaviors and phases of a cyberattack;
- we extended the FRVM, proposed by Sharma et al. [5], removing the assumption that the attack rate and the rate to trigger the MTD operation should be equal. It allows evaluating security metrics for more broad and realistic scenarios;
- we identify three key security metrics — Attack Success Probability (ASP), Mean Time to Compromise (MTC), and a Security Improvement metric (SI) — and evaluate them in two use-case scenarios where we consider SQL injection and Dictionary

attacks.

The remainder of this paper is structured as follows. §II provides background information regarding Software Defined Networks (SDNs), System security, security analysis and modeling, and the MTD technique of IP shuffling. §III presents a motivating example of an MTD-enabled SDN-based environment. §IV presents the proposed models for quantitative security analysis. §V demonstrates the applicability and feasibility of the proposed models, discussing their numerical results, trade-offs and limitations. §VI discusses related work on MTD for system and networks. §VII concludes this paper.

## II. PRELIMINARIES

### A. SDN-based systems

SDNs ensure first programmability and easy management of networks [9]. They separate data transmission from a network control layer, which administrators can use to shape traffic using a logically centralized controller rather than reaching out to all network components individually.

The above ease of configuration and maintenance has led to the wide adoption of SDN technology. They have become a core technology in cloud computing. Moreover, the centralized control over the network simplifies deploying defense mechanisms, such as MTD. However, one should note that MTD is not limited to SDNs and can as well be applied in traditional networks (e.g., through hardware-specific appliances), although at potentially higher operational costs [10].

### B. System security

Distinct reference models have mapped the cyber-attack phases as presented by *Mazurczyk and Caviglione* [11] to easily track the multi-stage characteristics of cyberattacks, showing that launching effective attacks requires gathering information about the victim network, a phase commonly referred to as *reconnaissance phase*. Cyber Kill Chain [8] identifies seven cyberattack phases: (1) Reconnaissance, (2) Weaponization, (3) Delivery, (4) Exploitation, (5) Installation, (6) Command and Control, and (7) Act on Objective. During (1), adversaries collect information about the network topology, service dependencies, and unpatched vulnerabilities [12]. Consequently, applying MTD mechanisms at the network level has a good chance of canceling the finding of such information by changing the system before the attack is deployed (in steps (3-5)). MTD can also help complicate subsequent steps of the Cyber Kill Chain, however, it remains crucial to implement changes as early as possible and before the adversary has time to begin with subsequent steps.

Various MTD techniques have been proposed as an attempt to increase the difficulty of reconnaissance [11], including IP address and port shuffling [5] (see §II-D). SDNs facilitate the implementation and deployment of such MTD techniques.

### C. Security Analysis & Modeling

The security of systems can be analyzed through measurements, simulation, and analytical modeling [13]. Analytical models, such as Petri nets [14], Markov chains [15] and the Hierarchical Attack Representation Model (HARM) [16] have been used for quantifying the security of systems and MTD mechanisms [17], [18], [16]. Their success motivated us to adopt an extension of Petri nets, called *Deterministic and Stochastic Petri Nets* (DSPN) [14] to evaluate services in SDN environments with and without MTD mechanisms.

DSPNs are ideal for modeling time-based MTD mechanisms since it allows deterministic transitions — that can be applied to trigger an MTD operation regularly — and stochastic transitions — that can be used to represent events with some time variance since they follow an exponential time distribution. DSPNs follow the regular Petri nets notation. Tokens are kept in places, and transitions consume tokens from a place generating new tokens in another place. We denote tokens by small black circles, places by white circles, and transitions by rectangles. Immediate transitions are thin black, stochastic transitions white, and deterministic transitions black bold rectangles. Arcs (represented by arrows) connect places and transitions. It rules the tokens flow through the places and can have weights. Inhibitor arcs (represented by an arrow ending with a small white circle) can disable a transition when its weight is met. We refer the reader to [14] and [19] for further details.

### D. MTD technique IP shuffling

IP shuffling [1] randomly changes IP addresses and port numbers to create a moving target of how components can be reached through the network. The shuffling frequency, which is the triggering interval for this MTD technique, crucially impacts security. For example, Sharma et al. [5] proposed FRVM to proactively change a host's vIP address and invalidate adversary knowledge on how the associated service can be reached. FRVM further demonstrates that the success probability for scanning attacks in static networks (without MTD) is at least 37% higher than in MTD-enabled networks. We will refer to this probability as the **Scanning Success Probability (SSP)**.

Let us provide further details on FRVM to the extent that they are relevant for this work. FRVM concludes that the SSP is determined by a hypergeometric distribution for static networks because it draws a vIP sequence of size $K$ from a finite population *without replacement*. For static networks, the SSP is therefore

$$P(X = x) = \frac{\begin{pmatrix} n \\ x \end{pmatrix}\begin{pmatrix} N-n \\ K-x \end{pmatrix}}{\begin{pmatrix} N \\ K \end{pmatrix}} \tag{1}$$

where an attacker has successfully obtained $x$ of the $n$ hosts in the address space $N \geq n$ by scanning the system $K \geq x$ times. This means the SSP to find at least one host in the static network is given by Eq. 2:

$$P(X > 0) = 1 - P(X = 0) = 1 - \frac{\begin{pmatrix} N-n \\ K \end{pmatrix}}{\begin{pmatrix} N \\ K \end{pmatrix}} \tag{2}$$

In an MTD-enabled network (using the FRVM), the SSP can be defined through a binomial distribution representing the number of successes when drawing a sequence of length $K$ from a finite population *with replacement* (see Eq. 3).

$$P(X = x) = \begin{pmatrix} K \\ x \end{pmatrix} p^x (1-p)^{K-x} \tag{3}$$

Here $p = \frac{n}{N}$ stands for the probability of an attacker discovering a host and $K \geq x$.

However, the FRVM, as an MTD approach, presents one strong assumption, namely that the attack rate and the MTD trigger rate should be equal. This assumption may affect the model accuracy since it would be reconfigured at runtime, collecting the current system attack rate and adapting the MTD operation interval. The proposed models in this paper extend FRVM by incorporating it and removing this assumption by allowing system representations where those rates are not necessarily equal.

## III. An MTD-enabled SDN-based System

We illustrate our approach with the help of an example SDN environment that employs a time-based IP shuffling

MTD, which we introduce in the following. Figure 1 shows our environment composed of an SDN controller (e.g., Ryu [20]), a domain name system (DNS server) to resolve virtual (*vIP*) and real (*rIP*) IP addresses of network components. The SDN further contains unmanaged switches ($SSW_1$, $SSW_2$, $USW_1$, and $USW_2$) and servers ($Server_1$ and $Server_2$) that offer *web-services* to users over the Internet.

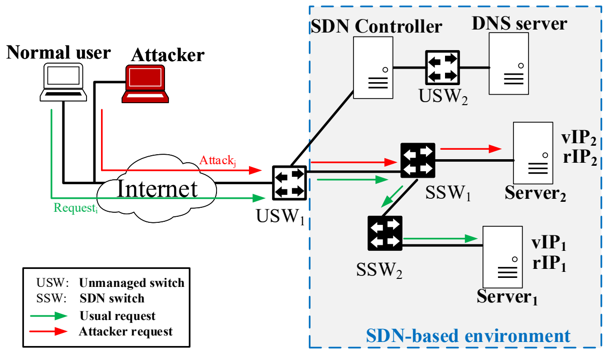


Fig. 1: An example of an SDN-based environment.

In this environment, we consider attackers trying to compromise a specific server (e.g., $Server_2$), requiring that the network systems protect $Servers_2$ from unauthorized access while guaranteeing service execution to regular users. We further assume that FRVM is applied as an MTD technique to mitigate cyberattacks. However, since FRVM adopts multiplexing and demultiplexing vIPs for the communication between network components, we also consider that the shuffling is applied for vIPs only. vIP shuffling can break active connections when changing a server's vIP. When the SDN controller performs vIP shuffling, it changes the switch flow rules. Consequently, ongoing attacks are disrupted since the attacker does not know the new vIP assigned to the server. In summary, we assume the following for this environment and the adversaries in it:

- the combination of [vIP: Port] or a domain name must be used to access a service. The domain name only redirects traffic to port 80 of a server;
- active connections are closed (broken) when a server's vIP gets shuffled in response to the MTD triggering;
- there is only one attacker in the system, and she/he does not scan the vIP twice unless the entire network address space has been already scanned;
- an attacker uses scanning software (e.g., Nmap [21]) for the reconnaissance phase, trying to identify valid vIPs;
- an attacker knows the size of the network and its subnets. The latter can, for example, be guessed using utilities like Whois [22];
- we condense weaponization, delivery, and exploitation into one phase and assume that, after an attacker has successfully exploited a host, it gains full control over it;
- the attacker persists in attacking the target system until she/he compromises at least one host.

## IV. Proposed Models

In this section, we introduce our models for quantitatively evaluating the security aspects of SDN-based systems, which employ IP shuffling as the MTD technique. First, we explain how the models work and how they extend FRVM. After that, we detail the metrics that can be computed using these models.

### A. Security Models

Our DSPN models represent attacker and system behavior in both a non-defending environment (i.e., w/o MTD) and in an environment where MTD is enabled using time-based vIP shuffling. They follow the defined environment constraints in §III. First, we present the model for an SDN without MTD and then for an SDN using MTD.

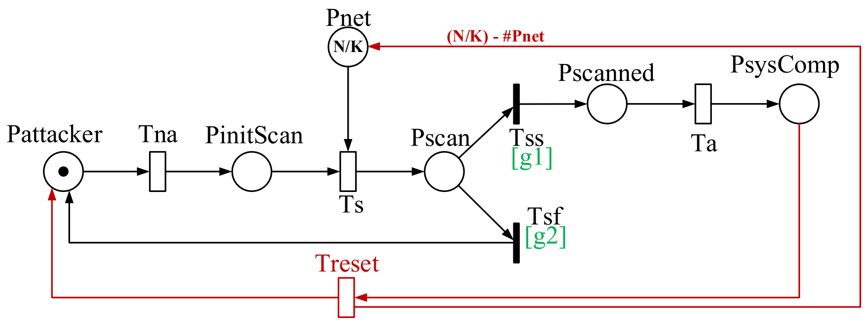


Fig. 2: DSPN model for a service under attacks in an SDN environment without deployed MTD.

*Model 1: No MTD.* This model represents an environment where no MTD mechanism is adopted. Figure 2 presents the proposed DSPN model for a non-defensive environment under attack. The presence of a token in the place `Pattacker` represents the attacker is idle and does not initiate the attack. This marking activates the exponential transition `Tna`, which represents the time the attacker waits to initiate the attack. When `Tna` fires, it generates a token in the place `PnitScan`, indicating the attacker can start the reconnaissance phase. In order to enable the transition `Ts` both `PinitScan` and `Pnet` places must have tokens. The place `Pnet` stores the number of tokens representing the maximum number of scans an attacker should perform to discover at least one valid vIP. $\frac{N}{K}$ represents this number since $N$ is the subnet size and $K$ is the number of scanning packets sent by the attacker (see §II-D). When `Ts` fires, it generates a token in place `Pscan`.

At this marking, both immediate transitions `Tss` and `Tsf` can be enabled. The guard functions — Boolean functions based on the current marking, disabling a transition from being fired when they return *false* — `g1` and `g2` (see Table I) and the weight functions (see Table II) determine which transition will be fired. The weight functions are based on FRVM model Eq. 2, where the result is the **SSP**, given by a number between 0 and 1. In this way, transitions `Tss` and `Tsf` receive different weights, representing, in this case, the firing probability of each one. The case of `Tsf` firing represents the attacker did not find a valid vIP, and a token is deposited in the place `Pattacker`, where the attacker waits some time and then initiates a new scanning process again. The case of `Tss` firing represents the attacker found a valid vIP, and can start to exploit vulnerabilities in the host. A token is generated into the place `Pscanned` to indicate this behavior.

TABLE I: Guard functions for the DSPN models.

| **Guard** | **Enabling Function** |
|---|---|
| ***g1*** | `(#Pscanned + #PsysComp) < 1` |
| ***g2*** | `(#Pnet > 0) OR (#Pnet=0) AND ((#Pscanned + #PsysComp) > 0))` |
| ***g3*** | `(#Pmtd = 1)` |
| ***g4*** | `(#Paa <` *DLI*`)` |

The presence of a token in the place `Pscanned` enables the transition `Ta`, representing the attacker starting the exploitation phase. Note that this transition can represent the time an attacker spends in a given attack (e.g., SQL injection), or it can be replaced to represent a more complex attack behavior (we demonstrate in §V-C how this can be done by representing a dictionary attack). When `Ta` fires, it generates a token in place `PsysComp`, indicating the vulnerability exploitation was successful,

TABLE II: Immediate transitions configurations adopted.

| Transition | Priority | Weight |
|---|---|---|
| ***Tss*** | 1 | IF(`#Pnet`=0): 0.99999 ELSE(1-(((($\#Pnet$ + 1) $\times$ $K$) - $K$) $\div$ (($\#Pnet$ + 1) $\times$ $K$))) |
| ***Tsf*** | 1 | IF(`#Pnet`=0): 0.00001 ELSE(((($\#Pnet$ + 1) $\times$ $K$) - $K$) $\div$ (($\#Pnet$ + 1) $\times$ $K$)) |
| ***Trc*** | 0 | 1 |
| ***Td1***, ***Td2***, ***Tdan***, ***Tdac***, ***Tna*** | 1 | 1 |

and the attacker gained access to the host. Transition `Treset` and the arcs in red are used only as an artifact to compute a metric through steady-state analysis. We detail this approach in the next Section §IV-B. Also, note that the command `#<place_name>` present in Tables I and II is used to capture the number of tokens in a place (e.g., `#PsysComp`).

*Model 2: MTD.* Figure 3 presents the proposed DSPN model for an environment using vIP shuffling as MTD. Different from the previous model, this one is divided into two submodels. Figure 3 (a) representing the *Clock Model* for the MTD mechanism and Figure 3 (b) representing the *Attacker and system model*. The *Clock Model* is responsible for controlling the frequency in which the vIP shuffling occurs (i.e., the MTD trigger interval). The presence of a token in the place `Pclock` enables the deterministic transition `Tt`. Unlike exponential transitions, where the firing delay varies following an exponential distribution, deterministic transitions fire within the specific delay when enabled. Thus, this type of transition represents the time-based behavior to trigger the MTD mechanism well. When `Tt` fires, a token is generated in the place `Pmtd`. This marking enables the guard function *g3* in the *Attacker and system model* (see Table I). Also, the transition `Trc` is enabled, but it will wait to fire until other transitions enabled by *g3* have fired because `Trc` has lower priority than other immediate transitions (see Table II).

The *Attacker and system model* works similarly to the no MTD model shown in Figure 2. However, when the guard function *g3* returns true, it activates the transitions `Tna` and `Td1`. The activation and firing of `Tna` means that place `Pnet` will have $\frac{N}{K}$ tokens again. It represents that the SDN controller assigned new vIPs to the network components. Consequently, the previous scans made by the attacker are not useful since the newer vIP could be in the range already scanned by the attacker. Thus, this model representation extends FRVM by resetting the SSP independently of the MTD or attack rate used.

When an attacker has already found a valid vIP in the network, and she/he is exploiting the host vulnerabilities, there is a token in the place `Pscanned` to represent this behavior. The enabling of `Td1` by *g3* removes the token from `Pscanned` and deposits a token in `Pattacker`. It means the attacker loses communication with the host and cannot proceed with the exploitation since it does not know the new vIP assigned to the host. Therefore, to successfully compromise a host, an attacker should find a valid vIP and exploit a vulnerability within the MTD trigger interval. Finally, similarly to the no MTD model, the transition `Treset` and arcs in red is only a modeling approach to compute a steady-state metric.

## B. Computed Metrics

Using the proposed models, we focus on computing three key security-related metrics: the Attack Success Probability (ASP), Mean Time to Compromise (MTC), and Security Improvement (SI). The ASP is computed through transient analysis, where it measures the probability of a token being in the place `PsysComp` over time. To obtain this metric, it *is not necessary* to use the red components of the model. MTC computes the mean time

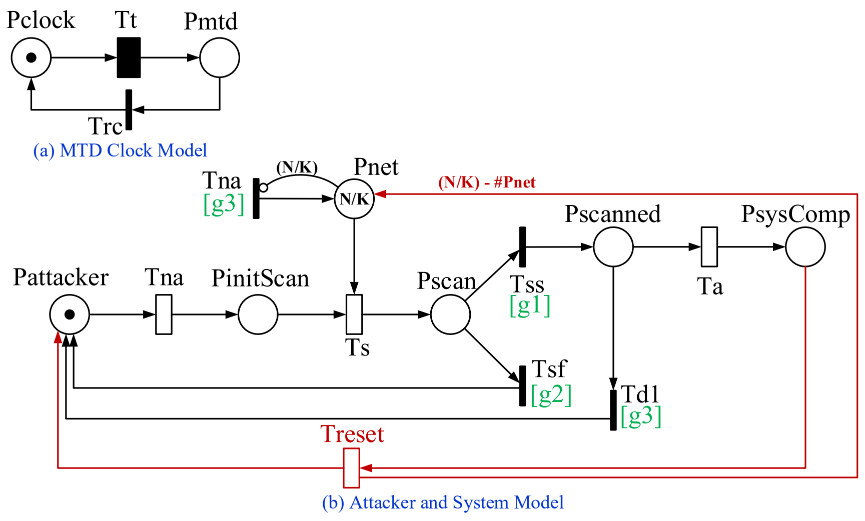


Fig. 3: DSPN model for an SDN adopting a time-based MTD under attacks.

for place `PsysComp` to receive a token, representing the attacker has compromised a host. This metric is obtained through stationary analysis. We use the models *with the red part* to get the result of this metric because every time a token arrives at the place `PsysComp`, it resets the token numbers of places `Pnet` and `Pattacker`, executing the model and get the result until a stationary state. We employ the Little's law [23] to calculate the MTC by dividing the expected number of tokens in the model ($N_{\mathrm{t}}$) by the throughput of the transition `Treset` ($\mathrm{TP}_{\mathrm{Treset}}$) as showed in Eq. (4). $\mathrm{TP}_{\mathrm{Treset}}$ is defined in Eq. (5), where $p$ is the probability of a marking that enables `Treset` and $T_r$ represents the delay assigned to `Treset`.

$$\mathrm{MTC} = \frac{N_{\mathrm{t}}}{\mathrm{TP}_{\mathrm{Treset}}} \tag{4}$$

$$\mathrm{TP}_{\mathrm{Treset}} = p \times \frac{1}{T_r} \tag{5}$$

The SI is obtained by calculating the percentage difference between the MTC result for a system without MTD and a system using MTD. Lastly, Table III details the expressions used to obtain the ASP and MTC using the proposed DSPN models. The function `P{<logical_expression>}` gives the probability of a marking occur (e.g., `P{#PsysComp > 0}`) and the function `E{#<place_name>}` indicates the expected number of tokens in a place (e.g., `E{#PsysComp}`).

TABLE III: Expressions used in the DSPN models to obtain the metrics results.

| **Metric** | **Expression** |
|---|---|
| ASP | `P{#PsysComp > 0}` |
| MTC | `(E{#Pattacker} + E{#PinitScan} + E{#Pscanned} + E{#PsysComp}) ÷ (P{#PsysComp > 0}/`$T_r$`)` |

## V. Case Studies

This section presents two case studies to demonstrate the applicability and feasibility of the proposed models. First, we present the experimental setup used to analyze the models. Next, we describe and present the first use case results, where we consider an attacker who wants to perform a SQL injection attack. Then, in the second use case, we show how to extend the proposed models to represent more complex attack behaviors, representing a dictionary attack scenario. We choose to represent these attacks because they are among the highly ranked web application attacks [24]. Lastly, we discuss the main results obtained and the implications of adopting the proposed models.

## A. Experimental Setup

We employed the TimeNET tool [25] to run and analyze the DSPN models. Other tools such as Mercury [26] or SHARPE [27] can be used similarly. Table IV describe the input parameters and their values assigned to the transitions of the DSPN models. Note that some parameters are only used in a specific use case. The parameters $T_{sc}$, $T_{sql}$, and $T_{dic}$ were obtained through experiments in an SDN testbed similar to the Figure 1. We used the Nmap [21] for scanning the network. Nmap was configured to send 10 TCP packets/vIP for a range of 4096 vIPs in the SDN. Then, we measured the total time spent to scan the vIPs and divided the result by 4096 to obtain the mean time to scan one vIP. We used the sqlmap [28] for automated SQL injection. The tool continually inserted malicious SQL queries into input data fields with the goal of retrieving valid user credentials of a server running a Damn Vulnerable Web Application (DVWA). We collected the mean time in which the tool has found the valid credentials. The Patator [29] tool was used to perform the dictionary attack. We passed a pre-defined dictionary to the tool, and configured the tool to attempt to log in to a web service using usernames and passwords from the dictionary. Then, we collected the mean time the tool could successfully login into the web service based on the valid credentials position index in the dictionary. The achieved mean values in all experiments are assigned to the respective input parameters in Table IV. Others parameters were reasonably estimated, but they can be easily adapted for the models.

https://github.com/digininja/DVWA

TABLE IV: Default input parameters for the DSPN models.

| Par. | Description | Transition | Value |
|---|---|---|---|
| $N$ | # subnet address space | - | 65536 (class C) |
| $K$ | # scanning vIPs | - | 4096 |
| *DLI* | Password location in the dic. | - | 10 |
| $T_{na}$ | Mean time to launch new attack | `Tna` | 60 s |
| $T_{sc}$ | Mean time to scan | `Ts` | 0.03557 s $\times$ $K$ |
| $T_{sql}$ | Mean time to exploit SQL inj. | `Ta` | 124.6406 s |
| $T_{dic}$ | Mean time to exploit dictionary | `Tda` | 4.87 s $\times$ *DLI* |
| $T_i$ | MTD interval | `Tt` | 120 s |
| $T_r$ | Aux. reset time | `Treset` | 0.01 s |

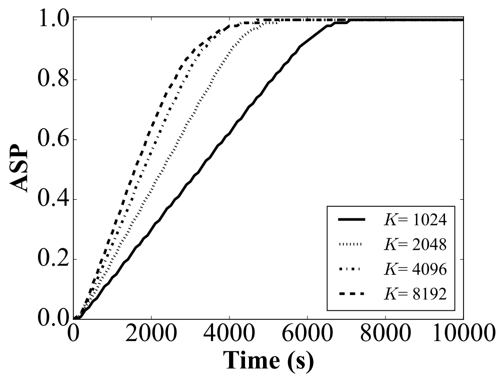
## B. CS#1. SQL injection representation

This case study aims to demonstrate the applicability of the proposed to compare security metrics between a system without and with MTD. In this case study, we consider an attacker who executes a SQL injection on a host. We use the DSPN models shown in §IV-A to represent this scenario. The SQL injection action is translated into the model by the transition `Ta`. So, the attacker first needs to perform a reconnaissance (i.e., scanning) in the network (`Ts`), find a valid host vIP (`Tss`), and then exploit the vulnerability by executing a SQL injection attack (`Ta`).

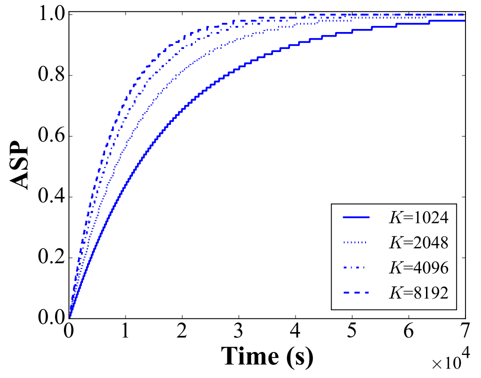
Figures 4a and 4b display the ASP obtained for this scenario by using the input parameters shown in Table IV and varying the parameter *K*. Recalling the parameter *K* represents the number of packets configured by the attacker to be sent during a scanning execution, Figures 4a and 4b show the ASP results for different values of *K*. We choose to vary this parameter because when executing scanning attacks, an attacker may attempt to avoid being detected by security systems, such as an Intrusion Detection Systems (IDS), sending fewer packets through the target network.

Through the plots, we can notice a considerable difference between the ASP for the system configuration of no MTD and with MTD. In the no MTD configuration (Figure 4a), the ASP reaches the maximum value after 5,000 seconds for most of the K values. The exception is for *K=1024*, when the attacker sends fewer packets, the maximum ASP is reached around $t$ = 7,000 seconds. Differently, in the MTD configuration (Figure 4b), the ASP only reaches the maximum value around the 50,000 seconds and around 70,000 seconds for the *K=1024*. It means the maximum ASP value is improved by around 10x.

We also computed the MTC for this scenario. Table V presents the MTC for the different values of K adopted.



(a) no MTD configuration



(b) MTD configuration

Fig. 4: ASP over time for the system without adopting the MTD (a), and with MTD enabled (b) considering different values for $K$ and $T_i$.

The MTC shows an attacker's average time to exploit and compromise a host successfully. Thus, these values do not necessarily relate to the time when the ASP reaches the maximum value, but it is more likely the MTC correspond with the time when the ASP reaches 0.5 - 0.65. A greater MTC means better security for the environment. The MTC results make clear the security improvements of deploying an MTD mechanism against scanning attacks. When calculating the SI, we can notice that an MTD mechanism enhances the environment security by 388% in the worst configuration (K = 8192) considering the adopted parameters.

TABLE V: MTC obtained considering different values for *K*.

| | MTC (s) | |
|---|---|---|
| **Config.** | **no MTD** | **w/ MTD** |
| *K = 1024* | 3,258.22 | 17,280.00 |
| *K = 2048* | 2,316.68 | 11,931.51 |
| *K = 4096* | 1,873.04 | 9,259.77 |
| *K = 8192* | 1,705.90 | 8,338.32 |

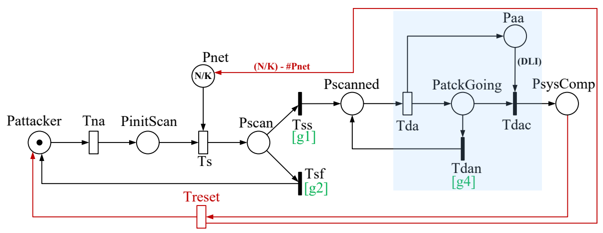
## C. CS#2. Dictionary attack representation

This case study demonstrates how it is possible to extend the proposed DSPN models to represent the behavior of different attacks. In this case study, we consider an attacker who executes a dictionary attacker in a host after finding it. A dictionary attack uses a list of predefined words with a correspondent hash to compare with the password hash, and when there is a hash matching, the attacker can identify the password [30]. In this use case, we consider an attacker executes the dictionary attack following a sequential order, from the first word to the $N$-st word, and the system has a weak password listed in the position *DLI* of the attacker's dictionary.

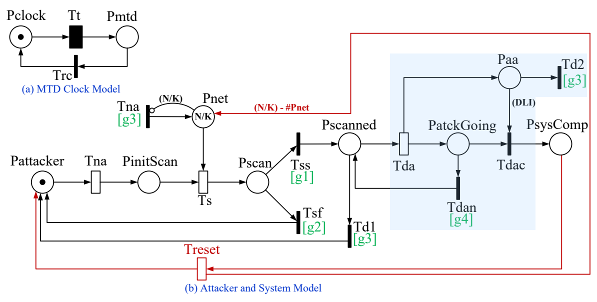
Figures 5a and 5b present the DSPN models to evaluate the proposed scenario of this use case, considering the environment without MTD and with MTD, respectively. The highlighted part (in blue) is the part that was extended from the original DSPN model of Figures 2 and 3. Note that the transition `Ta` in the original models was replaced by the highlighted part. This part represents the dictionary attack behavior. In this submodel, when an attacker has successfully scanned a host (token in the place `Pscanned`), she/he initiates executing the dictionary attacks (`Tda`) until the password position *DLI*. When the attacker has executed *DLI* attempts, the place `Paa` will have *DLI* tokens, and the place `PatckGoing` will have one token, enabling the transition `Tdac`, which means the attacker successfully exploited and compromised the host. The new guard function *g4* and the properties of the new immediate transitions are defined in Tables I and II, respectively. Note that since the net structure changed, the MTC metric expression should also be adapted to consider in the sum the expected number of tokens of the place `Paa` (i.e., `E{#Paa}`).

Next, we measured the ASP and MTC for the models using the parameters in Table IV. The plot in Figure 6 presents the ASP achieved over time for a system without MTD and when a system adopts a time-based MTD. Also, the plot shows the ASP when adopting two different MTD intervals ($T_i$). In the no MTD configuration, the ASP reaches 0.5 at t= 1,320 s and 1 at t= 3,400 s. On the other hand, in the MTD configuration, the ASP reaches 0.5 at t= 2,040 s using $T_i$ = 360 and at t= 3,360 s using $T_i$ = 120. The ASP reaches its maximum value (1) at t= 15,280 s using $T_i$ = 360 and at t = 25,500 s using $T_i$ = 120. Therefore, it is notable that the security improvements when adopting an MTD mechanism would be capable of delaying the system's compromise.

The analysis of the ASP over time is also valuable



(a) no MTD configuration



(b) MTD configuration

Fig. 5: DSPN model for system under a dictionary attack without adopting MTD (a) and using an MTD mechanism (b).

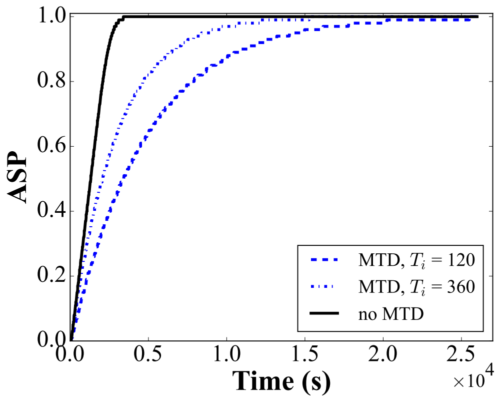


Fig. 6: ASP obtained for system under a dictionary attack considering the configurations (i) no MTD and (ii) adopting a time-based MTD with different MTD intervals.

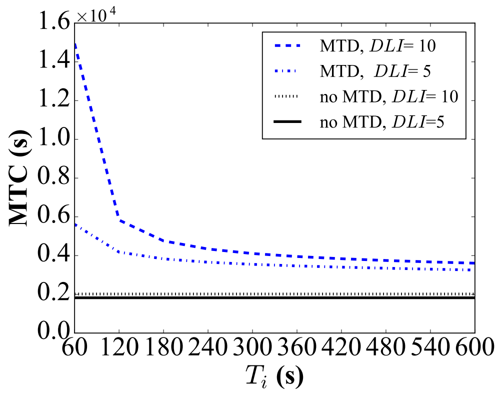
when defining the desired security levels on Service Level Agreement (SLA) contracts. Security engineers can analyze different configurations when using the proposed models to choose the most suitable for the

TABLE VI: Example of an ASP analysis considering different configurations for finding an ASP threshold.

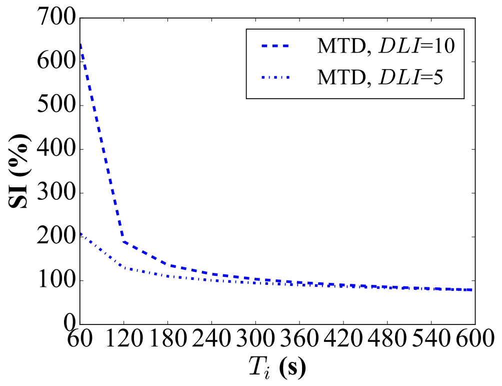
| | | w/ MTD, $T_i$ (s) | | |
|---|---|---|---|---|
| ***t (s)*** | **no MTD** | **120** | **240** | **360** |
| *600* | 0.21 | 0.1 | 0.15 | 0.17 |
| *1000* | 0.37 | 0.17 | 0.25 | 0.28 |
| *1400* | 0.54 | 0.24 | 0.33 | 0.38 |
| *1800* | 0.7 | 0.3 | 0.41 | 0.45 |

defined threshold. For instance, consider a system where the ASP should be kept below 30% for at least 30 min. Table VI presents an ASP analysis considering different configurations for the MTD interval. In the configurations analyzed in Table VI, the only configuration capable of keeping the ASP equal to 30% within 30 minutes is when adopting MTD with the trigger interval of 120 seconds.

Next, we analyze the MTC and SI for the scenario. Figure 7a presents the MTC achieved for the different configurations of $T_i$ and DLI, while Figure 7b shows the SI when using different MTD trigger intervals and assuming other values for the DLI. While assuming distinct values for the DLI have a negligible impact on the MTC when using the no MTD configuration, there is a more significant difference when an MTD mechanism is used. For instance, the MTC for the no MTD configuration are 1,821.74s (DLI = 5) and 2,016.57 s (DLI = 10), while for the MTD configurations the values reaches 5,610.7 (DLI = 5, $T_i$ = 60) and 14,943.85 (DLI = 10, $T_i$ = 60). The security enhancement is evident when we analyze the plot in Figure 7b. For the $Ti = 60$ the SI is more than 600% (DLI = 10) and 200% (DLI = 5) when compared with the environment without MTD. However, the SI is more marginal when using a $T_i > 300$ stabilizing near 100% and reaching 78% at $T_i = 600$ for both configurations of DLI.



(a) MTC results



(b) SI results

Fig. 7: MTC (a) and SI (b) analysis considering different MTD trigger intervals ($T_i$) and the password index in the attacker's dictionary (DLI).

Lastly, since we notice an MTC variation when assuming different values for the DLI, Figure 8 shows a plot analyzing the MTC for various values of DLI. In the scenario considered in this case study (sequential dictionary attempts), the DLI significantly impacts the MTC, especially when adopting an MTD mechanism. For the no MTD configuration changing the DLI from 5 to 25 increases the MTC by around 82%, and using the MTD 92% ($T_i$ = 360). When $T_i < (DLI \times T_{dic})$ the MTC grows exponentially, making it more hard for an attacker to compromise a system through a dictionary attack. This situation is shown for the DLI = 25 and $T_i$ = 120 in the plot of Figure 8, when the MTC reaches a value of 52,223.72 seconds.

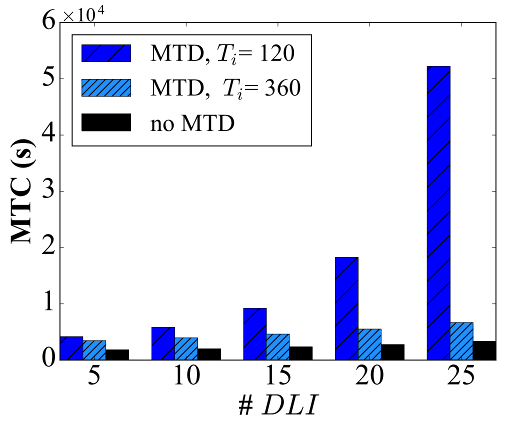


Fig. 8: MTC comparison between configurations by varying the DLI.

## *D. Discussion*

*Security enhancement.* The various analyses performed using the proposed DSPN models have shown that adopting a time-based MTD mechanism improves the security aspects of a system under cyber attacks that need a reconnaissance phase. The attack success probability (ASP) can drop significantly, as shown in the CS#1 (§V-B) where the maximum ASP value (1) dropped 10x if compared with the system without MTD. Also, the mean time to compromise a system (MTC) could be improved by at least 78% when an MTD mechanism was adopted (see §V-C). We also demonstrate the impact of assuming different MTD intervals. We could observe that the more significant security improvements in a dictionary attack scenario were when an MTD trigger interval ($T_i$) < 300 seconds was adopted.

*Models usability.* The case studies showed how the proposed DSPN models could help analyze systems regarding security aspects. Our proposed models that implemented and extended an existent MTD mechanism, agree with the initial FRVM model (refer to paper [5]), and the experimental analyses carried out by [31], showing that a time-based MTD can increase the security of an SDN environment. Additionally, our models relaxed a previous assumption on FRVM where the trigger interval and attack rate should be the same. Also, the adopted metrics can quantitatively evaluate different system configurations in order to achieve the desired security levels, which is especially useful for environments subject to SLA contracts. Besides, the case studies showed how security engineers could extend them to represent different types of attacks. However, in order to extend the models, one should comprehend DSPN modeling. Although it limits the models' usability, in future work, we can leverage modeling techniques that translate high-level models into Petri nets models, like in [32], which could facilitate the adoption of our proposed models.

*Security trade-offs.* Some research has been carried out regarding the trade-offs of adopting MTD mechanisms. Most of the works individually explored the security, performance, or availability impacts of MTD adoption. Few of those works could quantitatively analyze two of these aspects together. Quantitatively evaluating these three aspects of a system and correlating them together is a great challenge. Although the proposed models presented in this paper can be used to evaluate key security metrics, we understand that adopting an MTD mechanism can also impact the systems' performance and availability. In our recent paper [33], we evaluated the performance and availability aspects of systems adopting MTD. However, integrating security modeling increases models' complexity due to the different analyses that need to be carried out (e.g., transient and steady-state) to capture different metrics. Therefore, in future work, we will integrate the security models proposed in this paper with the performance and availability models to capture and analyze security, performance, and availability trade-offs.

## VI. Related work

Alavizadeh et al. [6] analyzed the metrics of risk, reliability, and unattackability when combining two different MTD techniques, shuffle and redundancy, in a cloud computing environment. The authors adopted Stochastic Reward Nets (SRN) and HARM models to conduct the analysis. They defend combining these two techniques to minimize risk while increasing system reliability. Similarly, El-Mir et al. [34] introduced a stochastic model to evaluate the security and downtime of virtual machines (VMs) in cloud computing environments. The models considered VM migration across different subnets as an MTD mechanism. Torquato et al. [35], [17] also presented SRN models to compute the attack success probability and availability of a system using VM migration scheduling as an MTD mechanism. Chen et al. [36] and Chang et al. [37] also employed SRNs to model and evaluate systems adopting MTD mechanisms regarding performance and security aspects. The first evaluated an MTD-enabled system concerning the job completion time, attack success probability, and system availability. The latter focused on analyzing only the mean job finish time of systems adopting an MTD mechanism.

Cai et al. [38] presented DSPN models to evaluate MTD mechanisms regarding performance. Their model could analyze performance metrics, such as the response time and system throughput. On the other hand, Maleki et al. [39] proposed a Markov model-based framework to analyze MTD-enabled systems. They introduced the concept of security capacity as a measure of MTD effectiveness and could estimate the probability of attack success and attack cost to evaluate the effectiveness of the proposed MTD mechanism. Connell et al. [12], [18] investigated system reconfiguration benefits when an MTD mechanism is applied in a pool of VMs. The focus of the works was to maximize the performance and availability of the VM pool. They developed an analytic model to evaluate the resource availability and performance when applying generic MTD mechanisms. The results found an optimal reconfiguration rate for the system concerning performance and availability constraints.

In a prior work [33], we used DSPN models, but to evaluate the performance and dependability of services in a network adopting a time-based MTD. Nguyen et al. [40] also evaluated the performance and dependability aspects of a system adopting a time-based MTD, but using SRN models. Their work provided helpful insights for our modeling strategy. In both works, authors measured performance and dependability metrics such as response time, throughput, host utilization, and availability. Sharma et al. [5] developed an MTD technique called FRVM to protect against network reconnaissance and scanning attacks. It allows a network component (e.g., a host) to have random vIP addresses multiplexed to its real IP address. The paper offered relevant insights into our model design and parameterization. In further work, Dishington et al. [31] used Mininet [41] to experimentally evaluate the deployment of FRVM in an SDN environment regarding security and performance aspects. The experiments demonstrated that IP address multiplexing was effective at further obfuscating and prolonging the network scans, but it comes with a significant performance loss.

The discussed works above mainly investigated the effectiveness of MTD mechanisms regarding security, performance, or dependability. However, none of them proposed extendable models capable of evaluating security key metrics such as ASP, MTC, and SI. Unlike the works above, this paper focuses on investigating the impact on systems' security introduced by deploying the vIP shuffling as a time-based MTD mechanism in an SDN environment. Therefore, our proposed models contribute to advancing the state-of-art on security modeling of MTD mechanisms. They are also suitable for identifying optimal configurations regarding distinct security system requirements based on their critical trade-offs.

## VII. Conclusions and Future Work

This paper presented a security modeling and evaluation approach for SDN environments adopting MTD mechanisms. We proposed a set of extendable and easily reusable DSPN models to capture the behavior of different cyberattacks in such an environment and analyze three key security metrics — Attack Success Probability (ASP), Mean Time to Compromise (MTC), and Security Improvement (SI). Using the proposed models, we evaluated two use cases considering highly ranked types of web-based attacks, SQL injection, and Dictionary attacks. The performed evaluation showed that the security significantly increased in both use cases thanks to the adoption of an MTD mechanism. Moreover, the use case evaluation also showed that the proposed models allow for flexible experimentation and tweaking the configuration of the non-MTD and MTD security schemes. This indicates that Petri Net modeling is perfect for security architecture trade-off analyses. Furthermore, in future work, we aim to advance our modeling approach by combining our security models with performance and dependability models to be able to evaluate the trade-off between these three characteristics.

## Acknowledgment

This research is supported by the Fond National de Recherche Luxembourg (FNR) through grant no. C20/IS/14689454/HERA.

## References

[1] T. E. Carroll, M. Crouse, E. W. Fulp, and K. S. Berenhaut, "Analysis of network address shuffling as a moving target defense," in *2014 IEEE International Conference on Communications (ICC)*. IEEE, jun 2014, pp. 701–706.

[2] E. Rhyne, "Moving Target Defense," Available at https://www.dhs.gov/science-and-technology/csd-mtd (2022/07/26).

[3] J. Rowe, K. N. Levitt, T. Demir, and R. Erbacher, "Artificial diversity as maneuvers in a control theoretic moving target defense," in *National Symposium on Moving Target Research*, 2012.

[4] J.-H. Cho, D. P. Sharma, H. Alavizadeh, S. Yoon, N. Ben-Asher, T. J. Moore, D. S. Kim, H. Lim, and F. F. Nelson, "Toward proactive, adaptive defense: A survey on moving target defense," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 709–745, 2020.

[5] D. P. Sharma, D. S. Kim, S. Yoon, H. Lim, J.-h. Cho, and T. J. Moore, "FRVM: Flexible Random Virtual IP Multiplexing in Software-Defined Networks," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 579–587.

[6] H. Alavizadeh, J. B. Hong, D. S. Kim, and J. Jang-Jaccard, "Evaluating the effectiveness of shuffle and redundancy mtd techniques in the cloud," *Computers & Security*, vol. 102, p. 102091, 3 2021.

[7] S. Alhozaimy and D. A. Menascé, "A formal analysis of performance-security tradeoffs under frequent task reconfigurations," *Future Generation Computer Systems*, vol. 127, pp. 252–262, 2 2022.

[8] E. M. Hutchins, M. J. Cloppert, R. M. Amin *et al.*, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues in Information Warfare & Security Research*, vol. 1, no. 1, p. 80, 2011.

[9] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[10] S. Sengupta, A. Chowdhary, A. Sabur, A. Alshamrani, D. Huang, and S. Kambhampati, "A survey of moving target defenses for network security," *IEEE Communications Surveys & Tutorials*, vol. 22, pp. 1909–1941, 2020.

[11] W. Mazurczyk and L. Caviglione, "Cyber reconnaissance techniques," *Communications of the ACM*, vol. 64, pp. 86–95, 3 2021.

[12] W. Connell, D. A. Menascé, and M. Albanese, "Performance Modeling of Moving Target Defenses," in *Proceedings of the 2017 Workshop on Moving Target Defense - MTD '17*. New York, New York, USA: ACM Press, 2017, pp. 53–63.

[13] O. C. Ibe, H. Choi, and K. S. Trivedi, "Performance evaluation of client-server systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 11, pp. 1217–1229, Nov 1993.

[14] M. Marsan and G. Chiola, "On petri nets with deterministic and exponentially distributed firing times," in *Advances in Petri Nets 1987*, 1987, vol. 266, pp. 132–145. [Online]. Available: http://dx.doi.org/10.1007/3-540-18086-9_23

[15] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1982.

[16] J. B. Hong and D. S. Kim, "Assessing the effectiveness of moving target defenses using security models," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 163–177, March 2016.

[17] M. Torquato, P. Maciel, and M. Vieira, "Analysis of vm migration scheduling as moving target defense against insider attacks," in *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, ser. SAC '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 194–202.

[18] W. Connell, D. A. Menasce, and M. Albanese, "Performance modeling of moving target defenses with reconfiguration limits," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, pp. 205–219, 1 2021.

[19] K. Trivedi and A. Bobbio, *Reliability and Availability Engineering: Modeling, Analysis, and Applications*. Cambridge University Press, 2017.

[20] Ryu project team, *RYU SDN Framework - English Edition*, ser. Release 1.0. RYU project team, 2014. [Online]. Available: https://books.google.lu/books?id=JC3rAgAAQBAJ

[21] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Sunnyvale, CA, USA: Insecure, 2009.

[22] L. Daigle, "WHOIS Protocol Specification," RFC 3912, Sep. 2004. [Online]. Available: https://www.rfc-editor.org/info/rfc3912

[23] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. Springer Publishing Company, Incorporated, 2010.

[24] O. Foundation, "Owasp top ten," 2021, [Online]. https://owasp.org/www-project-top-ten/. Accessed on 05/02/2021.

[25] A. Zimmermann, "Modelling and performance evaluation with timenet 4.4," in *Quantitative Evaluation of Systems*, N. Bertrand and L. Bortolussi, Eds. Cham: Springer International Publishing, 2017, pp. 300–303.

[26] B. Silva, R. Matos, G. Callou, J. Figueiredo, D. Oliveira, J. Ferreira, J. Dantas, A. Lobo, V. Alves, and P. Maciel, "Mercury: An integrated environment for performance and dependability evaluation of general systems," in *Proceedings of Industrial Track at 45th Dependable Systems and Networks Conference, DSN*, 2015.

[27] K. S. Trivedi and R. Sahner, "Sharpe at the age of twenty two," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 4, pp. 52–57, Mar. 2009.

[28] B. Damele and M. Stampar, "sqlmap," 2022, [Online]. https://sqlmap.org. Accessed on 20/06/2022.

[29] OffSec Services Limited, "Patator," 2022, [Online]. https://www.kali.org/tools/patator/. Accessed on 20/06/2022.

[30] E. Conrad, S. Misenar, and J. Feldman, "Chapter 6 - domain 5: Identity and access management (controlling access and managing identity)," in *CISSP Study Guide (Third Edition)*, 3rd ed., E. Conrad, S. Misenar, and J. Feldman, Eds. Boston: Syngress, 2016, pp. 293–327.

[31] C. Dishington, D. P. Sharma, D. S. Kim, J.-H. Cho, T. J. Moore, and F. F. Nelson, "Security and performance assessment of ip multiplexing moving target defence in software defined networks," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 8 2019, pp. 288–295.

[32] F. Machida, E. Andrade, D. S. Kim, and K. S. Trivedi, "Candy: Component-based Availability Modeling Framework for Cloud Service Management Using SysML," in *2011 IEEE 30th International Symposium on Reliable Distributed Systems*, Oct 2011, pp. 209–218.

[33] J. Mendonça, J.-H. Cho, T. J. Moore, F. F. Nelson, H. Lim, A. Zimmermann, and D. S. Kim, "Performability analysis of services in a software-defined networking adopting time-based moving target defense mechanisms," *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pp. 1180–1189, mar 2020.

[34] I. El Mir, A. Chowdhary, D. Huang, S. Pisharody, D. S. Kim, and A. Haqiq, "Software defined stochastic model for moving target defense," in *Proceedings of the Third International Afro-European Conference for Industrial Advancement — AECIA 2016*, A. Abraham, A. Haqiq, A. Ella Hassanien, V. Snasel, and A. M. Alimi, Eds. Springer International Publishing, 2018, pp. 188–197.

[35] M. Torquato, P. Maciel, and M. Vieira, "Security and availability modeling of vm migration as moving target defense," in *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*, vol. 2020-December. IEEE, 12 2020, pp. 50–59.

[36] Z. Chen, X. Chang, Z. Han, and Y. Yang, "Numerical Evaluation

of Job Finish Time under MTD Environment," *IEEE Access*, vol. 8, pp. 11 437–11 446, 2020.

[37] X. Chang, Y. Shi, Z. Zhang, Z. Xu, and K. Trivedi, "Job Completion Time under Migration-based Dynamic Platform Technique," *IEEE Transactions on Services Computing*, vol. 1374, no. c, 2020.

[38] G. Cai, B. Wang, Y. Luo, and W. Hu, "A model for evaluating and comparing moving target defense techniques based on generalized stochastic petri net," in *Advanced Computer Architecture*, J. Wu and L. Li, Eds. Singapore: Springer Singapore, 2016, pp. 184–197.

[39] H. Maleki, S. Valizadeh, W. Koch, A. Bestavros, and M. van Dijk, "Markov Modeling of Moving Target Defense Games," in *Proceedings of the 2016 ACM Workshop on Moving Target Defense - MTD'16*. ACM Press, 2016, pp. 81–92.

[40] T. A. Nguyen, M. Kim, J. Lee, D. Min, J.-W. Lee, and D. Kim, "Performability evaluation of switch-over moving target defense mechanisms in a software defined networking using stochastic reward nets," *Journal of Network and Computer Applications*, p. 103267, 11 2021.

[41] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: Association for Computing Machinery, 2010.