Quantifying the Security Effectiveness of Firewalls and DMZs

Huashan Chen
UT San Antonio
huashan.chen@utsa.edu

Jin-Hee Cho
US Army Research Lab
jin-hee.cho.civ@mail.mil

Shouhuai Xu
UT San Antonio
shxu@cs.utsa.edu

ABSTRACT

Firewalls and Demilitarized Zones (DMZs) are two mechanisms that have been widely employed to secure enterprise networks. Despite this, their security effectiveness has not been systematically quantified. In this paper, we make a first step towards filling this void by presenting a representational framework for investigating their security effectiveness in protecting enterprise networks. Through simulation experiments, we draw useful insights into the security effectiveness of firewalls and DMZs. To the best of our knowledge, these insights were not reported in the literature until now.

CCS CONCEPTS

• **Security and privacy → Distributed systems security**; **Network security**; **Firewalls**;

KEYWORDS

Firewalls, Demilitarized Zones, security metrics, security quantification, cybersecurity dynamics, preventive dynamics

ACM Reference Format:
Huashan Chen, Jin-Hee Cho, and Shouhuai Xu. 2018. Quantifying the Security Effectiveness of Firewalls and DMZs. In *HoTSoS '18: Hot Topics in the Science of Security: Symposium and Bootcamp, April 10–11, 2018, Raleigh, NC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3190619.3190639

1 INTRODUCTION

Firewalls and DMZs are two widely employed security mechanisms. On one hand, firewalls enforce security policies to filter or block unauthorized communication traffic between the outside network and an enterprise network or between the sub-networks within an enterprise network. The filtering operation can be conducted at the packet layer, the flow layer (i.e., examining flow-level content), and the application layer (i.e., inspecting application-layer data) [1, 8]. For the purpose of the present study, we focus on the functionality of firewalls in filtering unauthorized communication traffic, while safely assuming away the implementation details (e.g., the layers at which filtering is conducted). On the other hand, DMZs isolate the external network from an enterprise network while providing the external users with interfaces to access the enterprise's Internet-facing servers (e.g., websites and email servers). Intuitively, DMZs may slow down, or even prevent, some attacks against enterprise networks.

Despite the wide use of firewalls and DMZs, their security effectiveness has yet to be quantified and characterized. To the best of our knowledge, no prior studies have aimed at systematically answering the following question: *How much security is gained by employing firewalls and/or DMZs?* The void of this knowledge motivates the present study. The existence of the void is true despite that quantifying security is one of the well-recognized open problems [15, 18, 20]. Indeed, the importance of quantifying security has led to industrial efforts that focus on software vulnerabilities (e.g., the Common Vulnerability Scoring System or CVSS [17]) and academic investigations that treat an entire network as a whole (e.g., [4, 11, 20, 24]). However, the aforementioned motivational question remains unaddressed.

HoTSoS '18, April 10–11, 2018, Raleigh, NC, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-6455-3/18/04…$15.00
https://doi.org/10.1145/3190619.3190639

Our contributions. This work makes a first step towards quantifying the security effectiveness of firewalls and DMZs, by making two contributions. First, we propose a novel framework for modeling firewalls and DMZs in protecting enterprise networks, while treating software components as "atoms" in describing enterprise networks. Compared with the existing studies that aim to quantify security by treating an entire network as a whole, the present study have two salient features. (i) Existing studies often make the *independence* assumption between the attack events. For example, attacks against a victim (e.g., computer) are assumed to be independently waged by multiple compromised computers [2, 11, 24, 29, 30]. Although there have been some efforts to weaken the assumed independence [4, 26, 27], they can only accommodate some specific kinds of *dependence* rather than completely eliminating the matter of independence. The present study neither makes the independence assumption (unlike [2, 24, 29, 30]) nor assumes any specific kind of dependence (unlike [4, 26, 27]). We achieve this by developing a framework to allow for simulation studies, rather than for analytic treatment. (ii) The present study accommodates the threat models known as Lockheed Martin's Cyber Kill Chain [9] and Mandiant's Attack Life Cycle [14]. These threat models accommodate realistic attacks that are not considered in the existing studies mentioned above, which investigate epidemic spreading over *arbitrary* network structures [2, 24, 29, 30] or the more general notion of cybersecurity dynamics [7, 28, 31, 32].

Second, the framework guides us to conduct systematic simulation experiments. Our preliminary experiments lead to the following findings. (i) When the applications and operating systems (OSes) have few or too many vulnerabilities, firewalls and DMZ do not have a significant impact on security. This is because in the former case, the network cannot be attacked, with or without employing firewalls and DMZs, and in the latter case, these defense mechanisms cannot prevent attacks from succeeding. (ii) When the OSes are not vulnerable but the applications are, security effectiveness of firewalls and DMZs decreases as the fraction of vulnerable applications increases. (iii) When effective, employing perimeter firewalls alone has little impact on security, but further employing DMZ and internal firewalls (to separate an enterprise network into smaller ones) will substantially increase security. This justifies the practice of employing both DMZ and firewalls. (iv) When effective, employing perimeter firewall and DMZ increases security of the sever applications.

The rest of this paper is structured as follows. Section 2 describes the proposed framework. Section 3 presents the simulation experiments and the resulting insights. Section 4 reviews related prior studies. Section 5 concludes the paper with open problems for future research.

2 THE FRAMEWORK

The proposed framework consists of the following models to represent: (i) an enterprise network; (ii) vulnerabilities in the software stacks and vulnerabilities of human users; (iii) defense mechanisms to protect the network; and (iv) attacks against the network. The framework also includes security metrics to measure the outcome of attack-defense interactions. Due to space limit, we summarize the notations in Table 1 of the Appendix.

2.1 Representation of Networks

An enterprise network consists of n computers (including user computers and servers). The computers communicate with other computers in or outside the enterprise network.

2.1.1 Representation of Software Stacks. A computer runs a *software stack*, which has two layers: *application* and *OS*. An OS contains kernel functions and device drivers.

Representation of applications. An application consists of its own program code and the code of the software components upon which they depend (e.g., libraries). Each application is treated as an "atomic" entity because of the following: (i) each user process is an instance of an application; (ii) a vulnerable application can be an entry-point for remotely penetrating into a computer (e.g., remote code execution); (iii) an application is a privilege entity because if any part of an application is compromised, the entire application is compromised; and (iv) a system call from a malicious application can compromise the OS.

Let APP denote the universe of applications running in an enterprise network of n computers. For computer i in the network, we denote by $\text{app}_{i,z}$ the z-th application running on computer i, where $1 \leq i \leq n$ and $\text{app}_{i,z} \in \text{APP}$.

We classify applications into two types: clients (e.g., browsers, email clients) and servers (e.g., web servers, email servers, SQL servers). Server applications can be further divided into *Internet-facing* servers (i.e., accessible from the Internet) and *internal* servers (i.e., accessible from computers in the network but not from the Internet). We define the following mathematical function to represent this attribute:

$$\eta : \text{APP} \rightarrow \{0, 1, 2\} \tag{1}$$

such that '0' means client applications, '1' means Internet-facing server applications, and '2' means internal server applications. This classification is important because different kinds of applications play different roles in security. For example, a client application may be vulnerable to social engineering attacks (because the user may be less trained) while a server application may be not (because a server administrator may be better trained); an external attacker may directly compromise an Internet-facing server, but not an internal server unless the attacker already penetrated into the network.

Representation of OSes. An OS runs in the kernel space to manage computer hardware and software resources. We propose treating each OS function, rather than the entire OS, as an "atomic" entity because (i) applications often make system calls (*syscalls*) to invoke OS functions; and (ii) an OS is not compromised unless a vulnerable OS function is exploited (for example) by a *syscall* incurred by a malicious application.

Let OS denote the universe of OSes running in an enterprise network of n computers. For computer i in the network, $1 \leq i \leq n$, we denote by os_i the OS it runs and by $f_{i,z} \in \text{os}_i$ the z-th OS (kernel or device driver) function in os_i.

2.1.2 Representation of Computer i as $G_i = (V_i, E_i)$. We propose representing computer i as a graph

$$G_i = (V_i, E_i), \tag{2}$$

where G_i and E_i are defined below. For obtaining V_i, let $V_{i,app}$ denote the set of applications running on computer i and $V_{i,os}$ denote the OS of computer i. Then, we define

$$V_i = V_{i,app} \cup V_{i,os}. \tag{3}$$

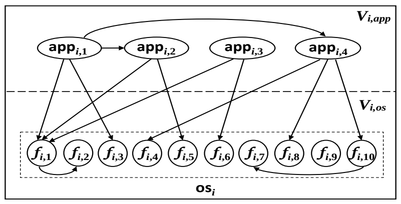


Figure 1: A graph-theoretic representation of computer i, $G_i = (V_i, E_i)$, in an enterprise network.

Fig. 1 shows a toy example of computer i, which runs four applications $\text{app}_{i,1}$, $\text{app}_{i,2}$, $\text{app}_{i,3}$ and $\text{app}_{i,4}$. The OS has 10 functions, namely $f_{i,1}, \ldots, f_{i,2}$. In this example, we have

$$V_{i,app} = \{\text{app}_{i,1}, \text{app}_{i,2}, \text{app}_{i,3}, \text{app}_{i,4}\}, \tag{4}$$

$$V_{i,os} = \text{os}_i = \{f_{i,1}, f_{i,2}, f_{i,3}, \ldots, f_{i,9}, f_{i,10}\}. \tag{5}$$

For obtaining E_i, we need to accommodate the security-related relations between the "atomic" entities mentioned above. We propose accommodating two kinds of relations, respectively dubbed *dependence* relation and *inter-application communication* relation. The *dependence* relation represents the *caller-callee* relation between two atomic software entities running on the same computer. There are two kinds of such relations: the caller-callee relation between applications and OS functions, denoted by an arc set $E_{i,af}$; the caller-callee relation between two OS functions (e.g., an application may make a syscall, which may further call another OS function), denoted by an arc set $E_{i,ff}$. The dependence relation should be accommodated because a vulnerability in an atomic entity on a caller-callee sequence can cause a successful exploitation. The *inter-application communication* relation, described by arc set $E_{i,aa}$, represents the communications between two applications running on the same computer. This relation should be accommodated because it can be exploited to wage attacks. For example, if one application is allowed to communicate with another application, the compromise of the former can cause the compromise of the latter assuming the latter has a vulnerability. In summary, we have:

- $E_{i,af}$ represents the *dependence* relation between the applications and the OS functions running on computer i. For example, in Fig. 1 we have $E_{i,af} = \{(\text{app}_{i,1}, f_{i,1}),$ $(\text{app}_{i,1}, f_{i,3}), (\text{app}_{i,2}, f_{i,1}), (\text{app}_{i,2}, f_{i,5}), (\text{app}_{i,3}, f_{i,1}),$ $(\text{app}_{i,3}, f_{i,6}), (\text{app}_{i,4}, f_{i,4}), (\text{app}_{i,4}, f_{i,8}), (\text{app}_{i,4}, f_{i,10})\}$.
- $E_{i,ff}$ represents the *dependence* relation between the OS functions running on computer i. For example, in Fig. 1 we have $E_{i,ff} = \{(f_{i,1}, f_{i,2}), (f_{i,10}, f_{i,7})\}$ because $f_{i,1}$ calls $f_{i,2}$ and $f_{i,10}$ calls $f_{i,7}$.
- $E_{i,aa}$ represents the *inter-application communication* relation between the applications running on computer i. For example, in Fig. 1 we have $E_{i,aa} = \{(\text{app}_{i,1}, \text{app}_{i,2}),$ $(\text{app}_{i,1}, \text{app}_{i,4})\}$ meaning that $\text{app}_{i,1}$ can initiate communications with $\text{app}_{i,2}$ and $\text{app}_{i,4}$ and that the compromise of $\text{app}_{i,1}$ can cause the compromise of $\text{app}_{i,2}$ and $\text{app}_{i,4}$, assuming the latter have vulnerabilities that can be exploited from $\text{app}_{i,1}$.

Then, we define

$$E_i \quad = \quad E_{i,af} \cup E_{i,ff} \cup E_{i,aa}. \tag{6}$$

2.1.3 Representation of the Inter-Computer Communication Relation within an Enterprise Network as Arc Set E_0. This relation describes which applications running on one computer can communicate with which applications running on another computer in the enterprise network. It should be accommodated because it reflects how attacks may be waged from a compromised computer to a vulnerable one, and because inter-computer communications are often monitored by firewalls.

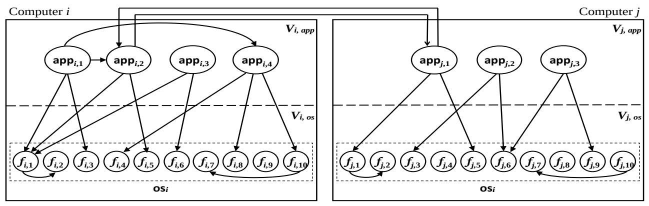


Figure 2: Illustration of the inter-computer communication relation $E_0 = \{(\text{app}_{i,2}, \text{app}_{j,1}), (\text{app}_{j,1}, \text{app}_{i,2})\}$.

Fig. 2 illustrates the inter-computer communication relation between computer i and computer j, which are respectively described by $G_i = (V_i, E_i)$ and $G_j = (V_j, E_j)$ as mentioned above. We use an *arc set* E_0 to represent the inter-computer communication relation between the applications running on computer i and the applications running on computer j, namely

$$E_0 \subseteq \{V_{i,app} \times V_{j,app}\} \cup \{V_{j,app} \times V_{i,app}\}, \tag{7}$$

where $1 \leq i, j \leq n$ and $i \neq j$. In the example illustrated in Fig. 2, suppose $\text{app}_{i,2}$ is a browser and is allowed to communicate with web server $\text{app}_{j,1}$. Then, we have

$$E_0 = \{(\text{app}_{i,2}, \text{app}_{j,1}), (\text{app}_{j,1}, \text{app}_{i,2})\},$$

where arc $(\text{app}_{i,2}, \text{app}_{j,1})$ reflects that vulnerable web server $\text{app}_{j,1}$ can be compromised by attacks that are waged from browser $\text{app}_{i,2}$, and arc $(\text{app}_{j,1}, \text{app}_{i,2})$ reflects that vulnerable browser $\text{app}_{i,2}$ can be compromised by malicious web server $\text{app}_{j,1}$ via, for example, the "drive by" download attack.

In general, we distinguish the aforementioned two kinds of attacks by partitioning E_0 into E_{00} and E_{01}, such that $E_0 = E_{00} \cup E_{01}$, E_{00} represents the attacks against clients (including peers in peer-to-peer application), and E_{01} represents the attacks against servers. More specifically, we have:

- $(\text{app}_{j,y}, \text{app}_{i,x}) \in E_{00}$ can be abused to launch attacks from a server or client or peer application $\text{app}_{j,y}$ against a client application $\text{app}_{i,x}$, where $\eta(\text{app}_{i,x}) = 0$.
- $E_{01} = E_0 \setminus E_{00}$: Any inter-computer communication other than what are accommodated by E_{00}.

We stress that $e \in E_0$ often corresponds to a communication or routing path, rather than a physical communication link.

2.1.4 Representation of the Internal-External Communication Relation as Arc Set E_*. The computers in an enterprise network often need to communicate with computers outside of the enterprise network. Similarly, computers outside of the network often need to communicate with computers in an enterprise network. Since these communications can be leveraged to wage attacks, we use the *internal-external* communication relation to accommodate them.

We use arc set $E_{*,io} = \{(\text{app}_{i,z}, *)\}$ to denote the *internal-to-external* communication relation from computer i to any external computer outside of the network, and use arc set $E_{*,oi} = \{(*, \text{app}_{j,z})\}$ to denote the *external-to-internal* communication relation from any computer outside of the network to computer j. Then, we define

$$E_* = E_{*,io} \cup E_{*,oi}. \tag{8}$$

We stress that $e \in E_*$ always corresponds to a communication path going through a number of routers.

2.1.5 Representation of an Enterprise Network as $G = (V, E)$. Putting together the pieces mentioned above, we represent a network of n computers as $G = (V, E)$, where

$$V = V_1 \cup \ldots \cup V_n \text{ and } E = E_1 \cup \ldots \cup E_n \cup E_0 \cup E_*. \tag{9}$$

For convenience, we may use $v \in V$ to indicate an arbitrary node v, and use $V_{(app)}$ and $V_{(os)}$ to respectively denote the set of applications and OSes running in the network, namely

$$V_{(app)} \quad = \quad V_{1,app} \cup \ldots \cup V_{n,app}, \tag{10}$$
$$V_{(os)} \quad = \quad V_{1,os} \cup \ldots \cup V_{n,os}. \tag{11}$$

2.2 Representation of Vulnerabilities

We consider two types of vulnerabilities [18]: *software vulnerabilities* and *human vulnerabilities*.

2.2.1 Representation of Software Vulnerabilities. Let VUL denote the set of software vulnerabilities in the software stacks of the computers in an enterprise network. We define a mathematical function

$$\phi : V \rightarrow 2^{\text{VUL}} \tag{12}$$

such that $\phi(v)$ represents the set of software vulnerabilities in node $v \in V$ (i.e., the software program running at node v). Note that $\phi(v) = \emptyset$ means that v is not vulnerable.

Since different kinds of software vulnerabilities can incur different consequences, each vulnerability $\text{vul} \in \text{VUL}$ has the following attributes:

- *Access required*: This attribute describes what kind of access an attacker must have in order to exploit $\mathsf{vul} \in \mathsf{VUL}$, namely whether or not the attacker has to have local access for exploiting vul. This attribute of vulnerabilities can be defined by predicate loc such that $\mathsf{loc}(\mathsf{vul}) = 0$ means that exploitation of vul requires local access and $\mathsf{loc}(\mathsf{vul}) = 1$ otherwise.
- *Zero-day*: This attribute describes whether a vulnerability $\mathsf{vul} \in \mathsf{VUL}$ is zero-day or not. This is important because the exploitation of a zero-day vulnerability often cannot be detected. We define predicate zd such that $\mathsf{zd}(\mathsf{vul}) = 0$ refers to known vul while $\mathsf{zd}(\mathsf{vul}) = 1$ indicates zero-day vul.
- *Privilege escalation*: This attribute describes the security consequence that can be caused by exploiting a vulnerability. We define priv such that $\mathsf{priv}(\mathsf{vul}) = 0$ means that the exploitation of vul will not grant the attacker the `root` privilege while $\mathsf{priv}(\mathsf{vul}) = 1$ means otherwise. This attribute is important because *remote-2-user* attacks [5, 6] exploit vul's with $\mathsf{loc}(\mathsf{vul}) = 1$ and $\mathsf{priv}(\mathsf{vul}) = 0$, *remote-2-root* attacks [10, 19, 21] exploit vul's with $\mathsf{loc}(\mathsf{vul}) = 1$ and $\mathsf{priv}(\mathsf{vul}) = 1$, and *user-2-root* attacks [6, 19, 23] exploit vul's with $\mathsf{loc}(\mathsf{vul}) = 0$ and $\mathsf{priv}(\mathsf{vul}) = 1$.

2.2.2 Representation of Human Vulnerabilities. Users of the computers may be subject to social engineering attacks. To model human vulnerabilities to social engineering attacks, we define mathematical function

$$\psi : V \longrightarrow [0, 1] \tag{13}$$

such that $\psi(v)$ for $v \in V_i$ represents the probability that the user of computer i is vulnerable to social engineering attacks.

2.3 Representation of Defenses

2.3.1 Representation of Firewalls. A firewall can monitor the inbound and outbound traffic of an enterprise network or the traffic between sub-networks of the enterprise network. A firewall has multiple physical *interfaces*, each corresponding to an internal sub-network or the external network [8]. If needed, multiple interfaces (e.g., multiple sub-networks) can be grouped into a *security zone* such that the traffic within a security zone is not monitored but the inbound and outbound traffic of a security zone is monitored [13]; otherwise, each interface corresponds to a security zone. The monitored traffic will be examined according to some security policies or rules, which specify what kinds of traffic are authorized (i.e., denial by default). For the purpose of the present study, firewalls are configured to only permit communications over the aforementioned *inter-application* communication relation E_0 and the *internal-external* communication relation E_*, which reflect the needs of the applications.

2.3.2 Representation of DMZs. DMZ is a security zone that typically hosts Internet-facing servers [25]. DMZ can isolate an enterprise network from the outside network by making external computers have no legitimate reasons to *directly* communicate with the computers in the enterprise network. For the purpose of the present paper, the implementation details of DMZs can also be safely assumed away. Nevertheless, we mention that a DMZ can be attained by using a firewall, which may have three kinds of interfaces: internal interfaces for connecting to the internal network, external interface(s) for connecting to the external network, and DMZ interface for connecting to a DMZ [22].

2.3.3 Representation of Other Defenses. We consider, in addition to firewalls and DMZs, the following defenses, while deferring the incorporation of others to future research.

HIPS. HIPS can enforce a *tight* or *loose* policy. A *tight* policy means that HIPS running on computer i monitors $V_{i,app}$ and E_i, which specifies the legitimate applications as well as which program entities are authorized to call which other entities. A *loose* policy means that the HIPS does not operate as such (e.g., the HIPS does not prevent an attacker with a user privilege (by compromising an application) running an arbitrary malicious program or making unauthorized calls to OS functions).

HIPS may be able to block privilege escalation attempts. Let ζ denote the probability that privilege escalation attempts are blocked by HIPS. It is realistic to assume that HIPS runs in the kernel space (i.e., HIPS is compromised when the OS is compromised) and that a compromised HIPS can behave arbitrarily (i.e., Byzantine). Since tracking $V_{i,app}$ and E_i for enforcing a tight policy is costly, we will identify conditions under which the defender does not have to make HIPS enforce a tight policy. This type of insights are useful because more efficient defense can be achieved without sacrificing security.

HIPS may also be able to block other attacks. Let α be the probability that any attack, other than privilege escalation, is blocked by HIPS.

NIPS. NIPS may be able to prevent attacks that attempt to exploit some *known* software vulnerabilities. Denote the number of *appearances* of known vulnerabilities by K in an enterprise network (e.g., 3 appearances of the same vulnerability in different computers leading to $K = 3$). In the ideal case, the defender should have patched all known vulnerabilities, meaning $K = 0$; in practice, some known vulnerabilities may not be patched. Suppose NIPS can block attacks that attempt to exploit a k fraction of the known vulnerabilities (i.e., $k \times K$ vulnerabilities cannot be exploited), where $0 \leq k \leq 1$.

2.4 Representation of Attacks

We describe attacks via two aspects: the set of *exploits* available to an attacker, and the *strategy* employed by the attacker. To describe attack consequences, we define

$$\mathsf{state}(v, t) : V \times T \longrightarrow \{0, 1\}$$

such that $\mathsf{state}(v, t) = 0$ means $v \in V$ is not compromised at time $t \in [0, T]$ and $\mathsf{state}(v, t) = 1$ means v is compromised at time t, where T is the time horizon of interest.

2.4.1 Representation of Exploits. Let X denote the set of exploits that are possessed by the attacker. We define

$$\rho : X \times \mathsf{VUL} \longrightarrow [0, 1] \tag{14}$$

such that $\rho(x, \mathsf{vul})$ represents the success probability when applying exploit $x \in X$ against vulnerability $\mathsf{vul} \in \mathsf{VUL}$.

In order to describe the exploitation capability of the attacker, let us denote by A the number of appearances of zero-day vulnerabilities in an enterprise network, by B the number of appearances of known vulnerabilities against which exploits can be blocked by the defender where $B = k \times K$, and by C the number of appearances of known vulnerabilities against which exploits cannot be blocked by the defender where $C = (1-k) \times K$. We represent the attacker's exploitation capability by (a, b, c), where a is the percentage of zero-day vulnerabilities that can be exploited by the attacker (i.e., the attacker has the exploits for $a \times A$ zero-day vulnerabilities), b is the percentage of known vulnerabilities that can be exploited by the attacker but the exploitation attempts are blocked by the defender (i.e., the exploitation of these $b \times B$ vulnerabilities will result in vein to the attacker), and c is the percentage of known vulnerabilities that can be exploited by the attacker without being blocked by the defender (i.e., the exploitation of these $c \times C$ vulnerabilities will succeed). The higher the a and c, the more attacks will succeed; the higher the b, the more attacks will be blocked.

2.4.2 Representation of Attack Strategy. For representing attack strategies, we use the attack lifecycle model highlighted in Fig. 3, which is adapted from Lockheed Martin's Cyber Kill Chain [9] and Mandiant's Attack Life Cycle [14]. The model includes six phases that are elaborated below.

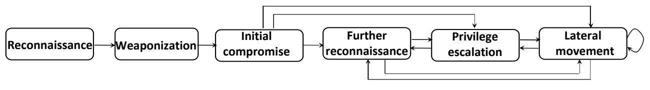


Figure 3: An attack lifecycle model adapted from Lockheed Martin's Cyber Kill Chain [9] and Mandiant's Attack Life Cycle [14].

Phase 1: Reconnaissance. Recall that an enterprise network is described by $G = (V, E)$, where V is the node set and E represents that legitimate communication relations. Reconnaissance means gathering information about a target enterprise network, including the communication topology G and the vulnerabilities in the software stacks of the computers in the network. The outcome of an reconnaissance process can be described by the attacker's view of the target network, denoted by $G' = (V', E')$, where $V' \subseteq V$ and $E' \subseteq E$ (i.e., G' is a sub-graph of G induced by the reconnaissance process). For each $v \in V'$, the attacker may further obtain information such as $\eta(v)$, namely the type of the application running at node v; $\phi(v)$, namely the set of software vulnerability v contains; and $\psi(v)$, namely the human factor vulnerability of v. Moreover, for each vulnerability $\mathsf{vul} \in \phi(v)$ where $v \in V'$, the attacker may further obtain information such as $\mathsf{loc}(\mathsf{vul})$, namely whether the vulnerability can be remotely exploited or not; $\mathsf{zd}(\mathsf{vul})$, whether the vulnerability is zero-day or not; and $\mathsf{priv}(v)$, whether the exploitation of the vulnerability can lead to a privilege escalation or not.

We propose using $\omega = |V'|/|V|$ to describe the attacker's *initial reconnaissance* capability. It is an interesting future study to investigate its alternate definitions, such as $|E'|/|E|$ or $(|V'| + |E'|)/(|V| + |E|)$.

Phase 2: Weaponization. Weaponization is for designing and developing a penetration plan according to the information gathered from the reconnaissance phase and the exploits possessed by the attacker. In particular, given the outcome $G' = (V', E')$ of the reconnaissance phase and the attacker's set of exploits X, the attacker now determines the nodes $v \in V'$ suitable for targets. Since initial compromises are often geared towards applications and there are two kinds of nodes in general (i.e., client application vs. server application), a candidate node for initial compromise should satisfy one of the following two conditions: one for client applications and the other for server applications.

On one hand, a candidate client node for initial compromise should satisfy the following conditions: (i) $v \in V_i$, where $V_i \subseteq V'$, runs a client application $\mathsf{app} \in \mathsf{APP}$ on computer i, namely $\eta(\mathsf{app}) = 0$; (ii) v is involved in some internal-external communication relation, meaning $(v, *) \in E_{*,io} \cap E'$ or $(*, v) \in E_{*,oi} \cap E'$; (iii) either app contains a software vulnerability, namely $\exists \mathsf{vul} \in \phi(v) = \phi(\mathsf{app})$, the app contains no vulnerability but an OS function called by the app contains a software vulnerable (i.e., there existing an access path from a secure app to a vulnerable OS function).

In order to precisely test the preceding condition (iii), we say that there is a *dependence path* between two nodes v and u in the *same* computer, say computer i or V_i, if there is, according to the *dependence relation* defined above, a path of dependence arcs starting from node v and ending at node u (i.e., the software program running at node u can be called, or reached, by the software program running at node v). We define the predicate

$$\mathsf{dep\_path}(v, u) : V_i \times V_i \longrightarrow \{\text{True}, \text{False}\} \tag{15}$$

such that $\mathsf{dep\_path}(v, u) = \text{True}$ if and only if there is a path of dependence arcs from v to u. As a result, the preceding condition (iii) can be formally described as

$$\begin{aligned} &(\exists \mathsf{vul} \in \phi(v), \exists x \in X : \psi(v) = 1 \wedge \rho(x, \mathsf{vul}) > 0) \vee \\ &(\exists \mathsf{vul} \in \phi(u), \exists x \in X : (u \in V_{i,os}) \wedge (v \in V_{i,app}) \wedge \\ &\quad \mathsf{dep\_path}(v, u) \wedge \psi(u) = 1 \wedge \rho(x, \mathsf{vul}) > 0)). \end{aligned} \tag{16}$$

Putting the preceding discussion together, we define the set of candidate client applications for initial compromise as:

$$\begin{aligned} \mathsf{Weapon}_0 = \{ &v \in (V' \cap V_{i,app}) : \eta(v) = 0 \wedge \\ &(((v, *) \in E_{*,io} \cap E') \vee ((*, v) \in E_{*,oi} \cap E')) \wedge \\ &\text{condition (16) holds} \}. \end{aligned} \tag{17}$$

On the other hand, a candidate server node $v \in V_{i,app}$ for initial compromise should satisfy the following conditions: (i) v runs an Internet-facing server application $\mathsf{app} \in \mathsf{APP}$ on computer i, meaning $\eta(\mathsf{app}) = 1$ and $(*, \mathsf{app}) \in (E_{*,oi} \cap E')$; (ii) either the app contains a remotely-exploitable software vulnerability, namely $\exists \mathsf{vul} \in \phi(\mathsf{app})$ such that $\mathsf{loc}(\mathsf{vul}) = 1$, or the app can call an OS function that contains a remotely exploitable vulnerability. Similar to the discussion above, the preceding condition (ii) can be precisely described as

$$\begin{aligned} &(\exists \mathsf{vul} \in \phi(v), \exists x \in X : \mathsf{loc}(\mathsf{vul}) = 1 \wedge \rho(x, \mathsf{vul}) > 0) \vee \\ &(\exists \mathsf{vul} \in \phi(u), \exists x \in X : (u \in V_{i,os}) \wedge (v \in V_{i,app}) \wedge \\ &\quad \mathsf{dep\_path}(v, u) \wedge \mathsf{loc}(\mathsf{vul}) = 1 \wedge \rho(x, \mathsf{vul}) > 0). \end{aligned} \tag{18}$$

Therefore, we can define the set of candidate server applications for initial compromise as:

$$\begin{aligned} \mathsf{Weapon}_1 = \{ &v \in V' \cap V_{i,app} : \eta(v) = 1 \wedge \\ &(*, v) \in (E_{*,oi} \cap E') \wedge \text{condition (18) holds} \}. \end{aligned} \tag{19}$$

By combining the two kinds of initial compromises, we obtain the set of candidates for initial compromise as:

$$\mathsf{Weapon} = \mathsf{Weapon}_0 \cup \mathsf{Weapon}_1. \tag{20}$$

Phase 3: Initial compromise. After determining Weapon according to (20), the attacker selects a subset of Weapon for initial compromise according to some strategy. An example strategy considered for our simulation study is:

- The attacker prefers using exploits against zero-day vulnerabilities to using exploits against known vulnerabilities. This is because exploits against zero-day vulnerabilities cannot be detected by the defense and using such exploits would enhance the attacker's chance in penetrating into the target enterprise network.
- The attacker strives to compromise the OSes whenever possible. This is because the compromise of an OS automatically causes the compromises of the applications running on top of it. This can also reduce the chance that the attack is detected when compared with the strategy that the attacker first compromises the vulnerable applications and then compromises the vulnerable OS beneath them (in this case, the attack against the applications may be detected by the defense because the defense may be employed in the kernel space).
- If the attacker cannot compromise the OS on computer i, then the attacker will strive to compromise all of the vulnerable applications on computer i.

According to this example strategy, the attacker can identify the set of suitable nodes for initial compromise, denoted by

$$\text{IniComp} = \{v \in \text{Weapon} : \text{attacker selects } v \text{ to attack}\}.$$

Phase 4: Further reconnaissance. Further reconnaissance means that once the attacker compromises a computer in the enterprise network, the attacker will attempt to obtain information about the sub-graph $G - G'$, where $G = (V, E)$ represents the enterprise network as well as the application-induced dependence relation, inter-application communication relation, and internal-external communication relation, and $G' = (V', E')$ is the sub-graph obtained by the attacker at the initial reconnaissance phase. Since further reconnaissance can be conducted recursively, we also use $G' = (V', E')$ to denote the outcome after further reconnaissance, while noting that $G' = (V', E')$ increases with further reconnaissance. This is so because, supposing $\text{app}_{i,1} \in V'$ (i.e., the attacker already knew obtained information about the node v at which application $\text{app}_{i,1}$ runs), arcs of the kind $(\text{app}_{i,1}, \text{app}_{j,2}) \in (E_0 \cap E')$ and the kind $(\text{app}_{m,2}, \text{app}_{i,1}) \in (E_0 \cap E')$ will be discovered by the attacker, and so are nodes $\text{app}_{j,2}$ and $\text{app}_{m,2}$, where $j, m \neq i$. Therefore, the attacker will update its information about the enterprise network as

$$\begin{aligned} V' &= V' \cup \{\text{app}_{j,2}, \text{app}_{m,2}\} \\ E' &= E' \cup \{(\text{app}_{i,1}, \text{app}_{j,2}), (\text{app}_{m,2}, \text{app}_{i,1})\}. \end{aligned}$$

Phase 5: Privilege escalation. After compromising an application $v \in V_{i,app}$ but not the underlying OS, the attacker would strive to achieve privilege escalation to compromise the underlying OS based on the outcome of further reconnaissance. This can be achieved by compromising some vulnerable OS functions. Another important factor affecting the success of privilege escalation is whether the HIPS enforces the aforementioned *tight* or *loose* policy.

When the HIPS enforces a *tight* policy with respect to E_i, the attacker cannot leverage a compromised application running at $v \in V_{i,app}$ to compromise the underlying OS via a vulnerable function running at $u \in V_{i,os}$ *unless* $(v, u) \in E_i$. More specifically, a privilege escalation occurs under the following condition:

$$\begin{aligned} &\exists v \in V_{i,app}, \exists u \in V_{i,os}, \exists \text{vul} \in \phi(u), \exists x \in X : \\ &\text{state}(v, t) = 1 \wedge \text{dep\_path}(v, u) \wedge \rho(x, \text{vul}) > 0. \end{aligned} \tag{21}$$

When the HIPS enforces a *loose* policy, the attacker can leverage a compromised application running at $v \in V_{i,app}$ to compromise the underlying OS via a vulnerable function running at $u \in V_{i,os}$ even if $(v, u) \in E_i$. More specifically, a privilege escalation occurs under the following condition:

$$\begin{aligned} &\exists v \in V_{i,app}, \exists u \in V_{i,os}, \exists \text{vul} \in \phi(u), \exists x \in X : \\ &\qquad \text{state}(v, t) = 1 \wedge \rho(x, \text{vul}) > 0. \end{aligned} \tag{22}$$

Phase 6: Lateral movement. Lateral movement means that after penetrating into an enterprise network, the attacker can leverage the inter-computer communication relation $e \in E'$ to attack other computers in the network. This can occur in one of the following two scenarios, depending on the new victim is a client (including a peer in peer-to-peer applications) or server application.

In the first scenario, the attacker has compromised a client or server or peer application on computer i and attempts to use the inter-computer communication relation $e \in (E' \cap E_{00})$ to compromise a client application on computer j. This occurs under one of the following two conditions:

$$\begin{aligned} &(\exists u \in V_{j,app}, \exists \text{vul} \in \phi(u), \exists x \in X : \text{state}(v, t) = 1 \wedge \\ &\qquad \text{state}(u, t) = 0 \wedge (v, u) \in (E' \cap E_{00}) \wedge \\ &\qquad \psi(u) = 1 \wedge \rho(x, \text{vul}) > 0); \end{aligned} \tag{23}$$

$$\begin{aligned} &\vee(\exists u \in V_{j,app}, \exists w \in V_{j,os}, \exists \text{vul} \in \phi(w), \exists x \in X : \\ &\qquad \text{state}(v, t) = 1 \wedge \text{state}(u, t) = 0 \wedge \\ &\qquad (v, u) \in (E' \cap E_{00}) \wedge \psi(u) = 1 \wedge \\ &\qquad \text{dep\_path}(u, w) \wedge \rho(x, \text{vul}) > 0). \end{aligned} \tag{24}$$

In the second scenario, the attacker has compromised a client or server application on computer i and attempts to use the inter-computer communication relation $e \in (E' \cap E_{01})$ to compromise a server application on computer j. This occurs under one of the following two conditions:

$$\begin{aligned} &(\exists u \in V_{j,app}, \exists \text{vul} \in \phi(u), \exists x \in X : \text{state}(v, t) = 1 \wedge \\ &\qquad \text{state}(u, t) = 0 \wedge (v, u) \in E' \cap E_{01} \wedge \\ &\qquad \rho(x, \text{vul}) > 0 \wedge \text{loc}(\text{vul}) = 1); \end{aligned} \tag{25}$$

$$\begin{aligned} &\vee(\exists u \in V_{j,app}, \exists w \in V_{j,os}, \exists \text{vul} \in \phi(w), \exists x \in X : \\ &\text{state}(v, t) = 1 \wedge \text{state}(u, t) = 0 \wedge (v, u) \in E' \cap E_{01} \wedge \\ &\qquad \text{dep\_path}(u, w) \wedge \rho(x, \text{vul}) > 0 \wedge \text{loc}(\text{vul}) = 1). \end{aligned} \tag{26}$$

Note that Eqs. (23) and (25) say that an application app running on computer j can be exploited from a compromised application running on computer i when app contains a software vulnerability; Eqs. (24) and (26) say that an application app running on computer j can be exploited from a compromised application running on computer i when app calls a vulnerable OS function.

2.5 Security Metrics

We define three metrics to describe the outcome of the attack-defense interactions: *percentage of compromised applications* at time

t or pca(t), *percentage of compromised server applications* at time t or pcsa(t), and *percentage of compromised OSes* at time t or pcos(t). Formally, we define:

$$\begin{aligned}
\text{pca}(t) &= |\{v \in V_{(app)} : \text{state}(v,t) = 1\}|/|V_{(app)}|, \\
\text{pcsa}(t) &= \frac{|\{v \in V_{(app)} \wedge \eta(v) \neq 0 : \text{state}(v,t) = 1\}|}{|\{v \in V_{(app)} \wedge \eta(v) \neq 0\}|}, \\
\text{pcos}(t) &= |\{v \in V_{(os)} : \text{state}(v,t) = 1\}|/|V_{(os)}|.
\end{aligned}$$

The security effectiveness of employing a specific combination of firewalls and DMZs is defined as $1 - \text{pca}(t)$, $1 - \text{pcsa}(t)$, and $1 - \text{pcos}(t)$, respectively. By comparing these metrics resulting from multiple deployments of different combinations of firewalls and DMZs, we can derive the relative security effectiveness between the deployments.

3 SIMULATION EXPERIMENTS

3.1 Simulation Setting and Methodology

3.1.1 Obtaining Representation of an Enterprise Network. We consider an enterprise network of 1,000 desktops and five servers. For the experimental study, we need a concrete $G = (V, E)$ representation of an enterprise network. As described in Section 2, we obtain a concrete $G = (V, E)$ as follows.

Software stacks. Suppose the five servers respectively run one of the following server applications: web server, email server, DNS server, FTP server, and database server. Suppose each desktop runs 4 applications: web browser, email client, instant messaging (IM, a peer-to-peer application), adobe reader. Suppose each desktop may run a FTP client application with probability p_1 and a database client application with probability p_2, where $0 \leq p_1, p_2 \leq 1$. This means that employees can use these applications.

For the OS layer, we assume the OS is Microsoft Windows with three components: OS kernel, subsystem drivers, and the hardware abstraction layer. We use this example scenario as we can obtain realistic parameters, namely that there are 2,500, 1,300, and 130 functions respectively corresponding to these components [3, 12]. This means $|V_{i,os}| = 3,930$.

Obtaining the representation of computers. Recall that computer i is represented by $G_i = (V_i, E_i)$ for $i = 1, \ldots, 1,005$. In order to obtain E_i, we assume there are no *inter-application communications* between the applications mentioned above. This is natural in the present setting but may not be true in general. In order to obtain the *dependence* relation within computer i, we assume for simplicity that each OS function is called, directly or indirectly, by each application with probability δ.

Obtaining the representation of inter-computer communication relation E_0. In order to obtain E_0, we make the following assumptions: a web browser in the enterprise network needs to communicate with the web server and the DNS server in the network; an email client needs to communicate with the email server to retrieve and send emails; a FTP client (if present) needs to communicate with the FTP server; a database client (if present) needs to communicate with the database server; an IM application needs to communicate with the other IM applications within the same sub-network; the web server needs to communicate with the database server. Since the email clients need to communicate with each other, this communication relation is reflected in E_0.

Obtaining the representation of internal-external communication relation E_*. In order to obtain E_*, we make the following assumptions: a web browser needs to communicate with the web servers that reside outside of the network; IM applications need to communicate with their peers outside of the enterprise network; email clients in the network need to send emails to, and receive emails from, the external network; Adobe readers need to open PDF files received from the external network; the Internet-facing servers (i.e., web server, email server, and DNS server) can be accessed by external computers.

3.1.2 Representation of Vulnerabilities. Suppose each application or OS function has at most one vulnerability. Let N denote the number of vulnerabilities that uniformly reside in N OS functions. Let β be the probability that every application contains a vulnerability. The attributes of a vulnerability $\mathsf{vul} \in \mathsf{VUL}$ is set as follows: If vul belongs to an OS function, we set priv(vul)=1; otherwise, priv(vul)=0. Let ϑ(vul) be the probability vul can be exploited remotely for any $\mathsf{vul} \in \mathsf{VUL}$, namely Pr(loc(vul) = 1), and τ(vul) be the probability that $\mathsf{vul} \in \mathsf{VUL}$ is zero-day, namely Pr(zd(vul) = 1).

For human vulnerabilities, we let $\psi(v) \in [0, 1]$ be the probability that a client application is vulnerable to social engineering attacks for every $v \in V_i$. Note that this holds for every $v \in V_i$ because V_i corresponds to computer i, meaning it is the user of computer i who may be subject to social engineering attacks.

3.1.3 Representation of Defenses. Since we focus on quantifying the security effectiveness of firewalls and DMZ, our simulation considers the five example combinations of firewalls and DMZ illustrated in Fig.4, which are respectively represented by $\gamma = 0, 1, \ldots, 4$. The other parameters representing defenses are as described in the framework and will be given in specific simulation scenarios.

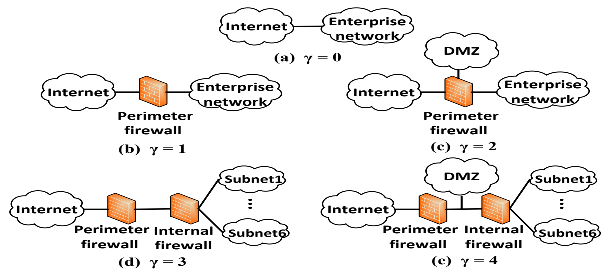


Figure 4: Five combinations of firewalls and DMZ employment (identified by $\gamma = 0, 1, 2, 3, 4$ and elaborated in the text).

Fig.4(a) corresponds to $\gamma = 0$, meaning neither firewall nor DMZ is employed. As a consequence, a communication not belonging to E_0 or E_* will not be blocked.

Fig.4(b) corresponds to $\gamma = 1$, meaning there is only a perimeter firewall to separate the enterprise network from the external network. That is, the firewall blocks any internal-external communication that does not belong to E_*.

Fig.4(c) corresponds to $\gamma = 2$, meaning that there is a perimeter firewall and a DMZ for running the Internet-facing servers (i.e., web server, email server, and DNS server). This firewall enforces as in the case of $\gamma = 1$ and further enforces that only the web server (both not the other two servers) in the DMZ can communicate with the database server in the internal enterprise network.

Fig.4(d) corresponds to $\gamma = 3$, meaning that there is a perimeter firewall and there are internal firewalls to separate the internal sub-networks from each other. The 1,005 computers are divided into six sub-networks, among which five sub-networks run 200 desktops each and the other sub-network runs the five servers. The perimeter firewall allows internal-external communications according to E_* and the internal firewalls allow inter-computer communications according to E_0.

Fig.4(e) corresponds to $\gamma = 4$, meaning that there is the same as in the case of $\gamma = 3$ except that the DMZ runs the Internet-facing servers (i.e., web server, email server, and DNS server). As in the case of $\gamma = 2$, the perimeter firewall enforces that only the web server (both not the other two servers) in the DMZ can communicate with the database server in the internal enterprise network.

3.1.4 Representation of Attacks. We consider an attacker outside of the network attempting to penetrate into the network and compromise as many computers as possible. Attacks proceed according to the strategy highlighted in Figure 3. The parameters reflecting attacks, namely ρ, ω and (a, b, c), will be given in specific simulation scenarios.

3.1.5 Simulation algorithm. The simulation algorithm executes the attach strategy discussed above. Due to space limit, we defer the pseudo-code of the simulation algorithm to the Appendix (Algorithm 1). Each simulation run leads to a sequence state(v, t) for $v \in V$ and $t = 1, \ldots, T$. We conduct 200 independent simulation runs with the given parameters specified below. From these sequences we derive the average values of metrics pca(t), pcsa(t) and pcos(t) per data point.

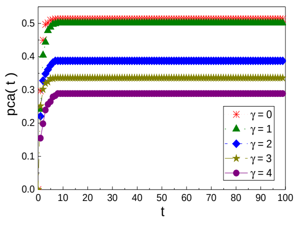
3.2 Simulation Results and Analysis

In the present simulation experiments, we assume that OSes are not vulnerable, but the HIPS and NIPS are not effective. This is because we focus on measuring the effectiveness of firewalls and DMZs. (In the extended paper, we will consider the other scenarios.) Because the OSes are not vulnerable, we only consider metrics pca and pcsa.

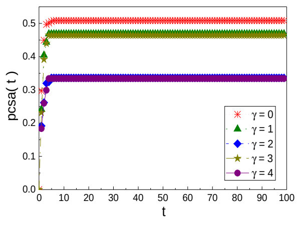
The parameters are set as follows: $p_1 = 0.1$ (the probability a desktop runs a FTP client application), $p_2 = 0.1$ (the probability a desktop runs a database client application), $\delta = 0.1$ (the probability that an OS function is called by an application), $N = 0$ (OSes are not vulnerable), $\psi(v) = 0.5$ (the probability $v \in V$ is vulnerable to social engineering attacks), $\vartheta(\mathsf{vul}) = 0.5$ (the probability vul can be exploited remotely), $\tau(\mathsf{vul}) = 0.5$ (the probability vul is zero-day), $k = 0$ (the probability an attack attempting to exploit a known-but-unpatched vulnerability is blocked by the NIPS), $\alpha = 0$ (the probability a social engineering attack is blocked by HIPS), $\zeta = 0$ (the probability a privilege escalation is blocked by HIPS), $(a, b, c) = (1, 1, 1)$, $\rho(x, \mathsf{vul}) = 1$ (the probability that $x \in X$ successfully exploits vulnerability $\mathsf{vul} \in \mathsf{VUL}$), $\omega = 1$ (the fraction of nodes that are discovered by the attacker's initial reconnaissance), and HIPS enforces the loose policy.

Note that the parameter setting implies the worst scenario in which the attacker can obtain all of the exploits against the vulnerabilities in the network and can discover all of the nodes via an initial reconnaissance. We consider $k = 0$ and $\alpha = 0$ (i.e., both NIPS and HIPS cannot block any attacks) because in the present paper we focus on measuring the security effectiveness of firewalls and DMZs.

Determining simulation time horizon T. Fig. 5 plots pca(t) and pcsa(t) of different combinations of firewalls and DMZ (reflected by $\gamma = 0, 1, 2, 3, 4$), where $\beta = 0.5$ (i.e., the probability that an application is vulnerable). We observe that both pca(t) and pcsa(t) first increase exponentially and then converge to a steady value. The exponential increase is caused by communications between email clients and/or between IM clients, namely that the compromise of any of these clients can cause the compromise of the other vulnerable clients. A similar phenomenon is observed for other values of β. Therefore, we will set $T = 100$ as the simulation time horizon for the simulation experiments reported below.



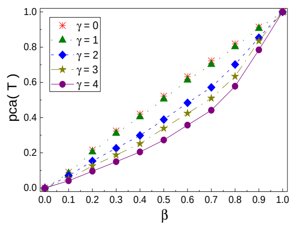
(a) pca(t)



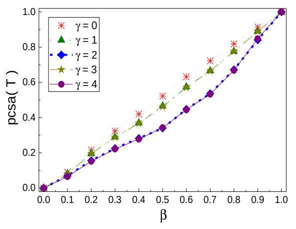
(b) pcsa(t)

Figure 5: pca(t) **and** pcsa(t) **of different combinations of firewalls and DMZ.**

Security effectiveness of firewalls and DMZ. Fig.6 plots pca(T) and pcsa(T) for $T = 100$ with respect to β under different combinations of firewalls and DMZ ($\gamma = 0, 1, 2, 3, 4$). Fig.7 plots Δpca(T, γ) and Δpcsa(T, γ) for $T = 100$ with respect to β, where $\Delta\mathrm{pca}(T, \gamma) = \mathrm{pca}_{T,\gamma=0} - \mathrm{pca}_{T,\gamma}$, $\Delta\mathrm{pcsa}(T, \gamma) = \mathrm{pcsa}_{T,\gamma=0} - \mathrm{pcsa}_{\gamma}$, and $\mathrm{pca}_{T,\gamma}$ and $\mathrm{pcsa}_{T,\gamma}$ respectively represent pca(T) and pcsa(T) with respect to a specific γ value. Note that Δpca$(T, 0) = 0$ because $\gamma = 0$, namely that neither firewall nor DMZ is employed, which is the baseline case. The key findings are discussed below.



(a) pca(T)

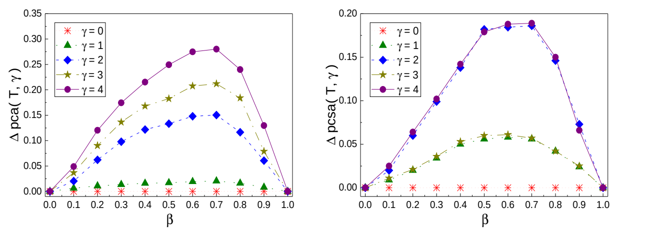


(b) pcsa(T)

Figure 6: pca(T) **and** pcsa(T) **with** $T = 100$ **and different** γ**'s (i.e., combinations of firewalls and DMZ).**

Finding 1: Fig. 6(a) shows that $1 -$ pca(100) with respect to a fixed firewall and DMZ configuration (i.e., fixed γ) decreases as β increases. This means that for a fixed defense strategy, the attack-defense interaction outcome is dominated by the degree of vulnerability of an enterprise network, namely the fraction of applications that contain at least one vulnerability. A similar phenomenon is also observed in Fig.6(b). This leads to:

INSIGHT 1. *When OSes are not vulnerable, the security effectiveness of a fixed combination of firewalls and DMZ decreases as the fraction of vulnerable applications increases.*



(a) $\Delta\text{pca}(T, \gamma)$ (b) $\Delta\text{pcsa}(T, \gamma)$

Figure 7: $\Delta\text{pca}(T, \gamma)$ and $\Delta\text{pcsa}(T, \gamma)$ with $T = 100$ and different γ's.

Finding 2: Fig.7 shows that $\Delta\text{pca}(T, \gamma)$ and $\Delta\text{pcsa}(T, \gamma)$ for a fixed γ are large when β is neither too small nor too large, but approaches zero when β tends to 0 or 1. That is, when few or most applications are vulnerable, firewalls and DMZ are not effective because in the former case, few applications can be compromised and in the latter case, most applications are eventually compromised.

Insight 2. *Firewalls and DMZ are not effective when few or most computers are vulnerable.*

Finding 3: Fig.7(a) shows that $\Delta\text{pca}(T, 1) \approx 0$ or 1.08% when averaged over β, meaning that the perimeter firewall is not effective. This is because the attacker can penetrate into the network via means that cannot be blocked by the perimeter firewall. When averaged over β, the difference between $\Delta\text{pca}(T, 1)$ and $\Delta\text{pca}(T, 2)$ is 8.71%, and the difference between $\Delta\text{pca}(T, 1)$ and $\Delta\text{pca}(T, 3)$ is 12.98%. Finally, we see that $\gamma = 4$ (i.e., enforcing comprehensive firewalls and DMZ defense) leads to the highest security among them, which justifies the real-world defense practice.

Insight 3. *Employing the perimeter firewall lone has a little security impact, but a comprehensive use of firewalls and DMZ can substantially increases security.*

Finding 4: Fig.7(b) shows that the security effectiveness firewalls and DMZ in terms of pcsa. We observe that $\Delta\text{pcsa}(T, 1) \approx \Delta\text{pcsa}(T, 3)$, meaning that employing perimeter firewall alone and employing both perimeter firewall and internal firewalls lead to the same security. This indicates that enforcing internal firewalls will not protect the server applications. However, $\Delta\text{pcsa}(T, 2) = 11.47\%$ when averaged over β, which highlights the security effectiveness of DMZ in protecting server applications. Nevertheless, $\Delta\text{pcsa}(T, 2) \approx \Delta\text{pcsa}(T, 4)$ affirms that security of server applications is protected by DMZ.

Insight 4. *Employing perimeter firewall and DMZ can substantially increase the security of sever applications.*

4 RELATED WORK

As discussed in the Introduction, quantifying security is one of the most fundamental open problems [15, 18, 20]. The present paper moves a further step beyond existing studies [4, 11, 24, 26–28] by getting rid of the independence assumption and by accommodating a new class of threats [9, 14]. To make the comparison fair, we should note that these gains in modeling capacity are obtained at the price of using simulations to characterize the security effectiveness of firewalls and DMZs. Nevertheless, the present study leads to new insights, such as those mentioned above and those will be reported in the full version of the present paper, that are not known until now. This can be attributed to the following fact: existing studies often use some parameters (e.g., probabilities) to abstract the capability of defense mechanisms such as firewalls and DMZs; in contrast, we aim to precisely quantify and characterize the security effectiveness of firewalls and DMZs by treating an entire enterprise network as a whole.

To the best of our knowledge, the present study initiates the investigation of how to quantify the security effectiveness of firewalls and DMZs. This can be justified by the fact that based on recent surveys [16, 18, 20], there are neither metrics nor models for explicitly measuring the security effectiveness of firewalls and DMZs by treating a network as a whole.

5 CONCLUSION

We presented a framework for quantifying the security effectiveness of firewalls and DMZs. The framework led to novel and useful insights. For example, when the applications and OSes have few or too many vulnerabilities, firewalls and DMZ do not have a significant impact on security; when the OSes are not vulnerable but the applications are, security effectiveness of firewalls and DMZs decreases as the fraction of vulnerable applications increases.

The present investigation can be extended in several directions, such as: (i) extending the simulation study to consider broader parameter regimes (e.g., the case of running multiple kinds of OS in enterprise networks); (ii) conducting case study to derive the structure $G = (V, E)$ of real-world enterprise networks; (iii) validating the proposed framework using real-world datasets; and (iv) examining more hostile scenarios in which firewalls can be compromised.

6 ACKNOWLEDGEMENT*

We thank the reviewers for their useful comments. This research was supported in part by the US Department of Defense (DoD) through the office of the Assistant Secretary of Defense for Research and Engineering (ASD (R&E)) and ARO Grant #W911NF-17-1-0566. The views and opinions of the author(s) do not reflect those of the DoD.

REFERENCES

[1] Ehab Al-Shaer, Hazem Hamed, Raouf Boutaba, and Masum Hasan. 2005. Conflict classification and analysis of distributed firewall policies. *IEEE journal on selected areas in communications* 23, 10 (2005), 2069–2084.
[2] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos. 2008. Epidemic thresholds in real networks. *ACM Trans. Inf. Syst. Secur.* 10, 4 (2008), 1–26.
[3] Geoff Chappell. [n. d.]. kernel-mode windows. https://www.geoffchappell.com/studies/windows/km/index.htm?tx=10. ([n. d.]).
[4] Gaofeng Da, Maochao Xu, and Shouhuai Xu. 2014. A New Approach to Modeling and Analyzing Security of Networked Systems. In *Proceedings of the 2014 Symposium on the Science of Security (HotSoS'14)*. 6:1–6:12.
[5] Rinku Dewri, Nayot Poolsappasit, Indrajit Ray, and Darrell Whitley. 2007. Optimal security hardening using multi-objective optimization on attack tree models of networks. In *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 204–213.
[6] Anup K Ghosh, Aaron Schwartzbard, and Michael Schatz. 1999. Learning Program Behavior Profiles for Intrusion Detection.. In *Workshop on Intrusion Detection and Network Monitoring*, Vol. 51462. 1–13.
[7] Yujuan Han, Wnelian Lu, and Shouhuai Xu. 2014. Characterizing the Power of Moving Target Defense via Cyber Epidemic Dynamics. In *Proc. 2014 Symposium on the Science of Security (HotSoS'14)*. 10:1–10:12.
[8] Ray Hunt. 1998. Internet/Intranet firewall security-policy, architecture and transaction services. *Computer Communications* 21, 13 (1998), 1107–1123.
[9] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. 2011. Intelligence-driven computer network defense informed by analysis of adversary campaigns

and intrusion kill chains. *Leading Issues in Information Warfare & Security Research* 1, 1 (2011), 80.

[10] Somesh Jha, Oleg Sheyner, and Jeannette Wing. 2002. Two formal analyses of attack graphs. In *Computer Security Foundations Workshop, 2002. Proceedings. 15th IEEE.* IEEE, 49–63.

[11] X. Li, P. Parker, and S. Xu. 2011. A Stochastic Model for Quantitative Security Analysis of Networked Systems. *IEEE Transactions on Dependable and Secure Computing* 8, 1 (2011), 28–43.

[12] Mateusz. [n. d.]. Windows WIN32K.SYS System Call Table. http://j00ru.vexillium.org/syscalls/win32k/32/. ([n. d.]).

[13] Alain Mayer, Avishai Wool, and Elisha Ziskind. 2000. Fang: A firewall analysis engine. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on.* IEEE, 177–187.

[14] Dan McWhorter. 2013. APT1: exposing one of China's cyber espionage units. *Mandiant. com* 18 (2013).

[15] David M. Nicol, William H. Sanders, and Kishor S. Trivedi. 2004. Model-Based Evaluation: From Dependability to Security. *IEEE Trans. Dependable Sec. Comput.* 1, 1 (2004), 48–65.

[16] Steven Noel and Sushil Jajodia. 2017. *A Suite of Metrics for Network Attack Graph Analytics.* Springer International Publishing, Cham, 141–176.

[17] The Forum of Incident Response and Security Teams FIRST. 2015. The Common Vulnerability Scoring System (CVSS). (June 2015). https://www.first.org/cvss

[18] Marcus Pendleton, Richard Garcia-Lebron, Jin-Hee Cho, and Shouhuai Xu. 2017. A survey on systems security metrics. *ACM Computing Surveys (CSUR)* 49, 4 (2017), 62.

[19] Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. 2012. Dynamic security risk management using bayesian attack graphs. *IEEE Transactions on Dependable and Secure Computing* 9, 1 (2012), 61–74.

[20] A. Ramos, M. Lazar, R. H. Filho, and J. J. P. C. Rodrigues. 2017. Model-Based Quantitative Network Security Metrics: A Survey. *IEEE Communications Surveys Tutorials* 19, 4 (2017), 2704–2734.

[21] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M Wing. 2002. Automated generation and analysis of attack graphs. In *Security and privacy, 2002. Proceedings. 2002 IEEE Symposium on.* IEEE, 273–284.

[22] Deb Shinder. [n. d.]. SolutionBase: Strengthen network defenses by using a DMZ. https://www.techrepublic.com/article/solutionbase-strengthen-network-defenses-by-using-a-dmz/. ([n. d.]).

[23] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on.* IEEE, 1–6.

[24] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos. 2003. Epidemic Spreading in Real Networks: An Eigenvalue Viewpoint. In *Proc. of the 22nd IEEE Symposium on Reliable Distributed Systems (SRDS'03).* 25–34.

[25] Avishai Wool. 2004. A quantitative study of firewall configuration errors. *Computer* 37, 6 (2004), 62–67.

[26] Maochao Xu, Gaofeng Da, and Shouhuai Xu. 2015. Cyber Epidemic Models with Dependences. *Internet Mathematics* 11, 1 (2015), 62–92.

[27] Maochao Xu and Shouhuai Xu. 2012. An Extended Stochastic Model for Quantitative Security Analysis of Networked Systems. *Internet Mathematics* 8, 3 (2012), 288–320.

[28] Shouhuai Xu. 2014. Cybersecurity Dynamics. In *Proc. Symposium on the Science of Security (HotSoS'14).* 14:1–14:2.

[29] Shouhuai Xu, Wenlian Lu, and Li Xu. 2012. Push- and Pull-based Epidemic Spreading in Arbitrary Networks: Thresholds and Deeper Insights. *ACM Transactions on Autonomous and Adaptive Systems (ACM TAAS)* 7, 3 (2012), 32:1–32:26.

[30] Shouhuai Xu, Wenlian Lu, Li Xu, and Zhenxin Zhan. 2014. Adaptive Epidemic Dynamics in Networks: Thresholds and Control. *ACM Transactions on Autonomous and Adaptive Systems (ACM TAAS)* 8, 4 (2014), 19.

[31] Ren Zheng, Wenlian Lu, and Shouhuai Xu. 2015. Active Cyber Defense Dynamics Exhibiting Rich Phenomena. In *Proc. 2015 Symposium on the Science of Security (HotSoS'15).* 2:1–2:12.

[32] R. Zheng, W. Lu, and S. Xu. 2017. Preventive and Reactive Cyber Defense Dynamics Is Globally Stable. *IEEE Transactions on Network Science and Engineering* PP, 99 (2017), 1–1.

APPENDICES

Table 1 summarizes the main notations used in the paper.

Algorithm 1 gives the pseudo-code of the simulation experiment.

APP	the universe of applications
OS	the universe of operating systems
η	η : APP $\to \{0, 1, 2\}$ indicates the types of applications: client ('0'), Internet-facing server ('1'), or internal server ('2')
G_i	$G_i = (V_i, E_i)$ is a computer, where $V_i = V_{i,app} \cup V_{i,os}$ and $E_i = E_{i,aa} \cup E_{i,af} \cup E_{i,ff}$
$G = (V, E)$	G is an enterprise network of n computers, where $V = V_1 \cup \ldots \cup V_n$ and $E = E_1 \cup \ldots \cup E_n \cup E_0 \cup E_*$
$G' = (V', E')$	G' is the attacker's view of the target network $G = (V, E)$ after reconnaissance process where $V' \subseteq V$ and $E' \subseteq E$
VUL	the universe of software vulnerabilities
$\phi(v)$	the set of vulnerabilities contained in node $v \in V$
$\psi(v)$	the probability that the user of computer i is vulnerable to social engineering attacks where $v \in V_i$
loc(vul)	whether the exploitation of vul $\in$ VUL requires local access ('0') or not ('1')
zd(vul)	whether vul $\in$ VUL is known ('0') or zero-day ('1')
priv(vul)	whether the exploitation of vul $\in$ VUL causes the attacker to get the root privilege ('1') or not ('0')
A	the number of zero-day vulnerabilities in G
K	the number of known vulnerabilities in G
B	the number of known vulnerabilities against which the exploits can be detected and blocked in G
C	the number of known vulnerabilities against which the exploits cannot be blocked in G
k	the fraction of known vulnerabilities can be prevented from being exploited by NIPS
α	the probability a social engineering attack is blocked
HIPS	the employment policy of HIPS
ζ	the probability privilege escalation attempts are blocked by HIPS
X	the set of exploits that are available to the attacker
$\rho(x, \text{vul})$	the probability $x \in X$ successfully exploits vul $\in$ VUL
(a, b, c)	the percentage of vulnerabilities that can be exploited by the attacker corresponding to (A, B, C)
ω	$\omega = \lvert V'\rvert/\lvert V\rvert$ is the fraction of nodes the attacker can discover by initial reconnaissance
p_1	the probability a desktop runs a FTP client application
p_2	the probability a desktop runs a database client application
δ	the probability that each OS function is called by each application
β	the probability that each application contains a vulnerability
N	the number of vulnerabilities in the OSes
$\vartheta(\text{vul})$	the probability vul $\in$ VUL is remotely exploitable
$\tau(\text{vul})$	the probability vul $\in$ VUL is zero-day
γ	the firewalls and DMZ employment
pca(t), pcos(t)	% of compromised applications and operating systems at time t, respectively

Table 1: Key notations and their meanings.

Algorithm 1 Simulation algorithm.

```
Input: enterprise network with (APP, OS, $p_1$, $p_2$, $\delta$); vulnerabilities with ($\beta$, $\vartheta$(vul), $\tau$(vul), $\psi$); defense with ($k$, $\alpha$, $\zeta$, HIPS); attacks with ($a$, $b$, $c$, $\rho$, $\omega$); simulation stop time $T$
Output: state($v$, $t$) for $v \in V$ and $t = 1, \ldots, T$
1: Generate simulation network $G = (V, E)$ with $\eta(v)$
2: Assign model parameters $\psi$, $\alpha$ to $v$, HIPS to $V_i \in V$
3: Simulate the reconnaissance
4: Weapon = $\emptyset$
5: for $v \in V'$ do
6:     if Eq. (20) holds for $v$ then
7:         Weapon = Weapon $\cup \{v\}$
8: Select IniComp according to Weapon
9: for $v \in V$ do
10:     state($v$, 0) = 0
11: for $v \in$ IniComp do
12:     Simulate initial compromise
13:     if $v$ is compromised then
14:         state($v$, 1) = 1
15: for $t \in \{2, \ldots, T\}$ do
16:     for each app $\in V_{(app)}$ with state($v$, $t - 1$) = 1 do
17:         Simulate further reconnaissance and update $G'$
18:         Simulate privilege escalation wrt Eqs. (21) or (22)
19:         Simulate lateral movement wrt Eqs. (23)-(26)
20: Return state($v$, $t$) for $v \in V$ and $t = 1, \ldots, T$
```