Analysis of Denial of Service Attack

Rakesh Rathi, Udai Nag

Abstract-- Proxy network-based defense has recently emerged to address an open research challenge. protecting Internet service applications from Denial-of-Service (DoS) attacks. Such schemes use a proxy network as a mediator for a hidden application to prevent direct attacks on the application's physical infrastructure, while maintaining communication between users and the application. The proxy network provides a distributed front-end to disperse DoS attack traffic, thereby shielding the application. However, the basic feasibility and fundamental properties of such schemes remain unclear, posing critical challenges for their use.

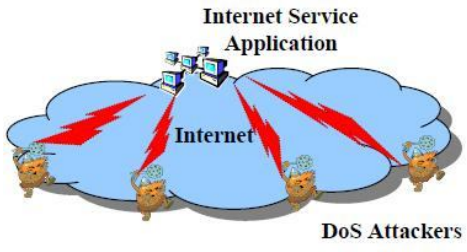
This Paper addresses these challenges by exploring proxy networks. Ability to resist important attacks: penetration, proxy depletion, and DoS attacks. We develop a generic analytic framework for proxy network-based systems, and use it to analyze proxy networks. resilience to penetration and proxy depletion attacks, characterizing how attacks, defenses, proxy network structure, and correlation in host vulnerabilities affect feasibility. Furthermore, using online simulation, we quantify the resistance to DoS attacks at an unprecedented scale and realism, by running real application, proxy network, and attack programs in a simulated network with a size comparable to tier-1 ISP networks.

We show that proxy network-based DoS defense can effectively resist these attacks, and protect applications successfully. Specific results are the following. First, proactive defenses, such as proxy migration, are required for penetration resistance . proxy networks can be effectively impenetrable with proxy migration, but will be penetrated easily without proactive defenses. Second, correlation in host vulnerabilities makes proxy networks vulnerable to penetration. By exploiting host diversity and intelligent proxy network construction, effective resistance can be achieved. Third, topology is crucial for resisting proxy depletion attacks: when a topology's eigenvalue is smaller than the speed ratio between defense and attack, all compromised proxies will always be recovered; when a topology's Palladian spectrum is larger than this ratio, compromised proxies will linger, making the proxy network unrecoverable. Last, proxy networks provide effective and scalable DoS defense. They can resist large-scale DoS attacks, while preserving performance for the majority (>90%) of users. Furthermore, increasing the proxy network size linearly improves the level of resistance to DoS attacks.

Keywords: Proxy Network, Denial of Service, Penetration, Defenses.

1. Introduction

DoS attacks are malicious attempts aiming to limit or deny service availability to legitimate users. A DoS attack on an Internet service application can be achieved by consuming critical resources (such as network bandwidth, server memory, disk space, or CPU time) on which the application or access to the application depends. Depletion of these resources can prevent the application from functioning, or disconnect the application from the Internet, and thus make the application unavailable to its users. A DoS attack occurs either at the infrastructure-level by attacking the resources directly (e.g. by flooding the application's sub-network with IP packets), or at the application-level by attacking through the application interface (e.g. by overloading the sufficient traffic to saturate even the largest Internet service applications. Therefore, such DoS attacks are a great threat to the availability of all Internet service applications.



application with abusive workload). In a typical DoS attack, an attacker first compromises a number of hosts (chosen from the hundreds of millions of vulnerable hosts) in the Internet, and then instructs these compromised hosts to attack an application by sending either infrastructure-level or application-level attack traffic to it (Figure 1.1). The recent emergence of sophisticated attacks tools, such as Trinoo [1], mstream [2], and TFN2K [3], and of Internet worms, such as CodeRed [4, 5], slammer [6], and MyDoom [7]. which automate the process of compromising hosts. makes it possible for attackers to control a large number (tens of thousands or even millions) of Internet hosts. These hosts can then be used to generate attack traffic, and to construct massive distributed DoS attacks, which can generate

Figure1.1:- Denial of Service Attacks

The real-world impact of these DoS attacks is severe. For example, in 1999, a series of large-scale DoS attacks targeted popular Internet service applications, such as Yahoo!, Amazon, eBay, and Buy.com [8,9]. These attacks kept the target sites offline for several hours, causing millions of dollars in lost revenue. In 2001, the .Code Red. and .Code Red II. worms spread widely in the Internet as part of a distributed DoS attack on the White House web site, forcing it to relocate [4]. In 2003, a series of large-scale DoS attacks using Internet

worms caused outages at Microsoft's website [6] and SCO Group's website [7]. According to a survey [10] of 251 organizations conducted by Computer Security Institute and the FBI, DoS attacks were the second-most costly computer crime, with damage exceeding 65 million dollars in 2003. These incidents and statistics show that DoS attacks have a serious economic and social impact.

Furthermore, DoS attacks are widespread in the Internet. In an attempt to characterize the frequency of DoS activities on the Internet, researchers at UCSD and CAIDA (the Cooperative Association for Internet Data Analysis) used backscatter detection techniques to infer DoS activities [18]. Their results reported more than 12,000 DoS attacks on more than 5000 targets during a span of three weeks, in February 2001. The victims of these attacks span the entire spectrum of commercial business sites, such as Yahoo!, CNN, as well as many small businesses. These numbers indicate that DoS attacks are common in the Internet, and that any Internet service application can become a victim of such attacks.

Since DoS attacks pose a critical threat to Internet service applications, researchers are exploring a wide range of defenses. As system researchers, our focus is infrastructure-level attacks, since these attacks target service infrastructures, and should be addressed at the system level. Application-level attacks are specific to the detailed structure of application interfaces, properties, and configurations, and thus can only be addressed by application designers. Existing system-level defense mechanisms [12-14] aim at blunting infrastructure-level DoS attacks1 by filtering the attack traffic. These schemes use routers to filter all the incoming network packets, and discard packets suspected to be part of an attack.

However, accurately distinguishing attack and normal packets is difficult, and increasingly so, as attack sophistication increases. As a result, these filter-based defenses are typically based on specific attack details, and do not apply generally to DoS attacks. For example, common methods use details of network packets, such as protocols (e.g. UDP or ICMP packets), the destination port, and source IP addresses [12-17], to identify attack packets. This lack of generality poses a fundamental limitation on their effectiveness.

Furthermore, in order for filter-based defenses to be effective, they must be deployed globally and in the basic Internet infrastructure of routers, since the attack traffic can come from millions of hosts dispersed across the Internet. Partial deployment leaves vast resources that can be used by attackers to generate devastating attack traffic which will saturate Internet service applications.

In summary, protecting Internet service applications from DoS attacks is a critical issue for Internet service applications. The current defense mechanisms are primarily based on filtering. They cannot protect applications from DoS attacks in general because they rely on specific attack details. Furthermore, they require global deployment with the basic Internet infrastructure. Due to these limitations, the filter-based defense mechanisms do not provide a general solution to the problem of protecting Internet service applications from DoS attacks.

2. Proxy Network Based DOS Defend

Recently, researchers have proposed the use of proxy networks as a system-level defense that protects Internet service applications from infrastructure-level DoS attacks [18-22] [23-27]. This new scheme does not suffer from the limitations of existing DoS defense mechanisms, and has shown promise in protecting applications. availability from DoS attacks. It is an attractive approach for DoS defense.

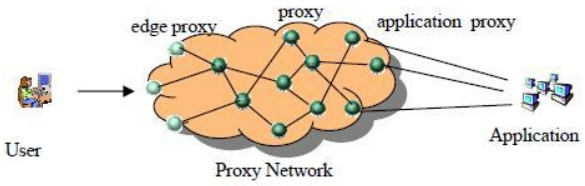


Figure1.2:- Proxy Network Based DOS Defend

A proxy network (Figure 1.2) is an overlay network composed of interconnected proxies which run on hosts dispersed across the Internet. In a proxy network-based DoS defense, a proxy network is used as an application mediator, delivering application messages between the application and its users. As shown in Figure 1.2, on one side of the proxy network, a set of proxies (known as application proxies) are connected to the application; on the other side of the proxy network, a select set of nodes (known as edge proxies) publish their IP addresses, providing application access to users.

Proxy network-based DoS defense is based on two key ideas. First, a proxy network mediates application messages between users and the application, providing the only public interface for application access. Since the proxy network delivers only application messages, this

prevents direct infrastructure-level DoS attacks on the application. Second, the proxy network presents a broad public access by using a large number of edge proxies. This broad front disperses the attack traffic, and dilutes the impact of even distributed DoS attacks.

Proxy network-based DoS defense has shown promise in accomplishing these key ideas, for the following reasons. First, an application is protected by a series of proxy indirections, all of which must be compromised by attackers to expose the application to direct attacks. Because the number of indirections can be adjusted by reconfiguring the proxy network, it provides a flexible structure for resisting an attacker's penetration and protecting the application from direct attacks. Second, the edge proxies can be widely dispersed, making it difficult for attackers to saturate all of them, and thereby, interrupt application service. This allows proxy networks to tolerate DoS attacks by dispersing attack traffic. By mediating application access to prevent direct attacks, and by providing a DoS-resilient front-end for the application to dilute the impact of attacks, a proxy network has the potential to protect the application from DoS attacks.

Furthermore, besides its potential to protect an application's availability, a proxy network-based DoS defense has shown promise for large-scale deployment. Since proxy networks are application-level overlay networks built on top of the Internet, they do not require any modification to the existing Internet infrastructure. This greatly facilitates large-scale deployment of proxy networks. Success of large-scale proxy networks, such as Content Delivery Networks (e.g. Akamai [29] proxy network which has over 15,000 proxies deployed in over 1,200 networks across 65 countries), demonstrates the practical feasibility of large-scale deployment of proxy networks.

In short, proxy network-based DoS defense is an attractive scheme for protecting Internet service applications from DoS attacks. It does not have the limitations of the existing DoS defense mechanisms. By mediating application access to avoid direct DoS attacks, and by providing a distributed front-end to shield the application from DoS attacks, a proxy network-based DoS defense shows promise in protecting an application's availability from DoS attacks. Furthermore, it is feasible to deploy a proxy network-based DoS defense scheme at the Internet-scale, providing a global DoS defense for Internet service applications in practice. Thus, this scheme has the potential to provide a feasible solution to protect Internet service applications from DoS attacks.

In order to study a general class of proxy networks, we develop a generic framework which encompasses a wide range of proxy network-based DoS defense. The framework defines key components of a proxy network system, and describes how attacks and defenses change the system state. It enables rigorous study of a large class of proxy networks, with results that bear on the entire class. Based on the generic framework for proxy network schemes, we develop a stochastic model to characterize how attacks and defenses change the state of system components quantitatively, thereby allowing for a rigorous study of system dynamics as a function of attacks and defenses. This generic framework and stochastic model provides a basis for our study of both penetration attacks and proxy depletion attacks.

A) Resistance to Penetration Attacks

Based on the generic framework and stochastic model, we combine analysis with Monte Carlo simulation techniques to study how long it takes a penetration attack to penetrate a proxy network. We study when a proxy network can resist penetration attacks for a long period of time, making such attacks practically impossible. We also study the impact of key system parameters on a proxy network.s resistance to penetration attacks, and identify the key system requirements for achieving effective defense.

B*)* Resistance to Proxy Depletion Attacks

We use the generic framework and stochastic model described earlier to characterize the impact of proxy depletion attacks on a proxy network system. Based on the framework and model, we study system dynamics as a function of attacks and defenses. We analyze when a proxy network can remove all the compromised proxies, regardless of how many proxies are compromised initially. This way, we characterize the circumstances when a proxy network can resist proxy depletion attacks effectively, and when it cannot. From these results, we develop guidelines for proxy network design.

C) Resilience to DoS attacks

We study the properties of proxy networks under DoS attacks empirically, using online packet-level network simulation with full applications, a real software implementation of proxy network, and real attacks. In particular, our experiments are performed using a large-scale online simulator. MicroGrid [30] which enables packet-level accurate simulation of large-scale network environments with 10,000 routers and 40 Autonomous

Systems (ASes). These network sizes are comparable to a large ISP network. Furthermore, Microgrid supports direct execution of unmodified application binaries, and thus allows us to use real applications and a real proxy network implementation in the simulation. In our study, we build a DDoS zombie network (comparable to one which contains 10,000 zombies with DSL or cable modem connections) with a real DoS attack toolkit [8], and use the zombies to generate attack traffic. Total attack traffic intensities up to 6.4Gbps, and a wide range of DoS attack scenarios are explored.

This experimental configuration is large and real enough to capture key properties of the Internet environment and application dynamics, such as router queues, packet drops, real temporal and feedback behavior of network and application protocols, which are critical to the application behavior and performance under DoS attacks. Therefore, this approach enables accurate modeling of the full complexity of network and application behavior needed to reproduce DoS dynamics, and to characterize application and proxy network performance in varied attack scenarios. With this leverage, we study application performance delivered by a proxy network for a range of proxy network structures and attack scenarios.

2.1 Denials-of-Service Attacks

A DoS attack is characterized by an explicit attempt to prevent legitimate users of a service from using that service. A DoS attack on an Internet service application can be achieved by consumption of scarce, limited, or non-renewable resources on which the application (or access to the application) depends. These resources may include network bandwidth, server memory, disk space, CPU time, and access to other computers and networks. Depletion of these resources can prevent the application from functioning or disconnect the application from the Internet, thereby causing service disruption and, thus, making the application unavailable to its users.

The impact of DoS attacks is severe. For example, DoS attacks have shut down high-profile sites, such as Yahoo!, Amazon, EBay and Buy.com [8,9], causing millions of dollars in lost revenue. A range of DoS attacks in recent years [4-7] disrupted the websites of the government and high-profile companies (such as Microsoft and sco.com), causing a significant social impact. According to a survey [10] collected from 251 organizations, DoS attacks were the second-most expensive computer crime, with a cost of more than 65 million dollars, in the year 2003.

Furthermore, DoS attacks are a widespread phenomenon in the Internet. For example, studies [11] reported more than 12,000 DoS attacks on more than 5000 targets during the short span of three weeks in February 2001. The victims of these attacks span the entire spectrum of commercial business sites, such as Yahoo!, CNN and many small businesses.

In conclusion, DoS attacks are a major threat to Internet service applications. They are widespread in the Internet, threaten the availability of various Internet service applications, and cause significant economic and social impact. Therefore, protecting Internet service applications from DoS attacks is an important problem.

In the following, we first classify DoS attacks according to their high-level approaches because each approach presents a unique set of problems; then, we describe how DoS attacks are constructed.

2.2 Classification of Denial-of-Service Attacks

DoS attacks on an Internet service application can be achieved either by directly attacking the resources on which the application (or access to the application) depends, or by attacking through the application interface. We classify DoS attacks as *infrastructure-level* and *application-level* attacks, according to these high-level approaches. Infrastructure-level attacks target the service infrastructure resources directly, such as the networks and hosts of the application; for example, by sending floods of network traffic to saturate the victim network, attackers can disconnect the application from its users. In contrast, application-level attacks exploit an application's weaknesses via the application interface; for example, by overloading the application with an abusive workload, attackers can make the application unavailable to legitimate users.

Infrastructure-level and application-level DoS attacks are fundamentally different. Infrastructure-level attacks focus on the service infrastructure resources (e.g. hosts and network), regardless of the application running on that infrastructure; the details of the application are irrelevant to such attacks. In contrast, application-level attacks focus on the weaknesses of the application, regardless of the service infrastructure the application uses; the details of the application are critical to these attacks.

This distinction makes defense against infrastructure-level and application-level DoS attacks fundamentally different problems. The key challenge in

defending against infrastructure-level attacks is building a system to protect the service infrastructure. In contrast, the key challenge in defending against application-level attacks is making an application robust. Since each application is unique, this is an application-specific problem, and there are no system-level solutions. As system researchers, we focus on infrastructure-level DoS attacks and explore system-level solutions that protect Internet service applications from infrastructure-level DoS attacks. We leave application-level DoS attacks for application designers to solve.

3. A Generic Framework for Proxy network based DOS Defense

Researchers explore the use of proxy networks as mediators to protect Internet applications from DoS attacks [18-22, 28]. Two key elements are the common core of all of these approaches (e.g. SOS [18, 19] and i3 [21, 22, 28]). First, all these approaches use an overlay network . proxy network . to mediate communication between users and applications. As long as the application is only accessible via the proxy network, the application servers cannot be attacked directly. Second, all these approaches use a large set of public proxies to provide access to the application and allow the number of public proxies to be increased flexibly. In order to deny application service, attackers must saturate this large number of proxies. The flexibility enables scalable resilience against DoS attacks. The commonality of these approaches allows them to be studied within a single framework.

In this section, we propose a generic framework which captures the key elements of all proxy network approaches and defines a system state model which describes the impact of attacks and defenses. The framework serves two purposes: 1) it provides a formal basis for discussion of proxy networks and attacks, and 2) it enables study of properties of a large class of proxy networks. We use this framework to study both penetration attacks and proxy depletion attacks. In the following, we introduce our generic framework, and then discuss how previously proposed proxy network schemes are captured in the framework.

3.1 Definition of the Generic Framework

The framework for proxy network schemes has two parts, a description of system components, including applications, users, hosts, and a generic proxy network, and a description of how attack and defense processes affect system dynamics.

3.1.1 System Components

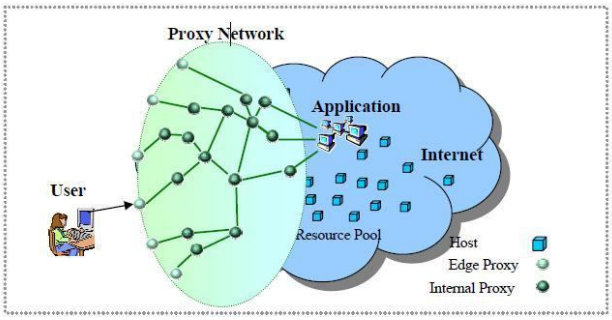


Figure 3.1 Generic Frameworks for Proxy Networks

As shown in Figure 3.1, our generic framework describes a system where a proxy network mediates all traffic between an application and its users, and protects the application from infrastructure-level DoS attacks. In the following section, we define the four key system components: applications, users, hosts, and a proxy network.

A) Application

An application is a deployed software system that implements an Internet service which responds to user requests and runs on a host in the Internet. In the proxy network scheme (see Figure 3.1), the IP address of the application is hidden and the application has connections with the proxy network, through which the application communicates with its users.

B) Users

A user is the principal that uses the application client software to interactively access the application, in order to use the application service. For example, a user can be a person using a web browser to access the Internet service application. In the proxy network scheme (see Figure 3.1), users are outside the proxy network and access the application via edge proxies (defined below) and through the proxy network.

C) Hosts

A host is a computer system connected with the Internet which provides the software and hardware infrastructure to support the operation of proxy nodes (defined below). A large number of such hosts dispersed widely in the Internet form a resource pool for the proxy network (see Figure 3.1).

Hosts may have vulnerabilities, such as exploitable bugs in the operating system software, which allow attackers to compromise the hosts. Furthermore, the vulnerabilities of the hosts in the resource pool may be

correlated (e.g. same operating system software with the same bugs). If host vulnerabilities are correlated, once a host is compromised, others may be easily compromised using similar techniques.

D) Proxy Network

As shown in Figure 3.1, a proxy network is an overlay network which runs on the resource pool of Internet hosts and mediates all traffic to and from the application. A proxy network is a set of interconnected proxies, each of which is a software program that runs on an Internet host and forwards application traffic. There are two types of proxies, edge proxies and internal proxies. Edge proxies have published IP addresses. Internal proxies are those which are not edge proxies; their IP addresses are hidden.

As shown in Figure 3.1, on one side of the proxy network a selected set of proxies are connected to the application, and on the other side of the proxy network, a set of edge proxies publish their IP addresses providing access to users of the application. As such, the proxy network mediates all traffic between users and the application.

There are three important properties of a proxy network: *topology*, *depth,* and *width.*

The *topology* of a proxy network characterizes the internal connectivity amongst proxies. The topology of a proxy network can be represented by a graph, where vertices represent proxies and edges represent the connections among proxies. Technically two proxies are connected if they can route packets to each other. In the context of network security, the important distinction is that connected proxies know each other.s IP address.

The *depth* of a proxy network is the minimum number of proxy indirections between an application and its users. The depth of a proxy network for an application is defined as the minimum path length in the proxy network topology graph from any edge proxy to the application. For example, the depth of the proxy network shown in Figure 3.1 is four.

The *width* of a proxy network is the number of public access points the proxy network presents to the users of an application. The width of a proxy network is defined as the number of edge proxies. For example, the width of the proxy network shown in Figure 3.1 is six.

3.1.2 System Dynamics

System dynamics describes the changes in system state which result from attacks and defenses. By studying the system dynamics of a proxy network under various attack and defense scenarios, we can understand when the proxy network can provide stable defense against penetration attacks and proxy depletion attacks. We first introduce terminology to describe the system state, and then discuss how attacks and defenses affect the overall system dynamics.

A) System State

We define the state of system components as follows. A host has two states: compromised and intact. A host is compromised when attackers have control over it and any information stored there may be revealed to attackers. A host is intact if and only if it is not compromised.

A proxy has three states: exposed, compromised and intact. A proxy is exposed if attackers know its location, i.e. the IP address of the host where the proxy runs; in this case the proxy is subject to future attacks. A proxy is compromised if it runs on a compromised host. A proxy is *intact* if it is neither exposed nor compromised.

B) Attacks

Our generic framework captures a range of attacks, among which we study penetration attacks and proxy depletion attacks.

The goal of penetration attacks is to discover the IP address of the application protected by a proxy network. The strategy is to explore the structure of the proxy network and compromise proxies along a path in the proxy network towards the application. As shown in Figure 3.2, these attacks allow attackers to penetrate into the proxy network, reducing the distance between the application and the exposed proxies, and perhaps, eventually discovering the IP address of the application.

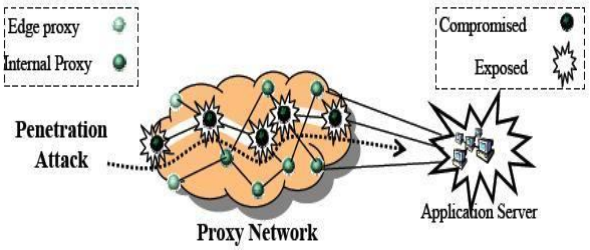


Figure 3.2 Penetration Attacks

The goal of proxy depletion attacks is to compromise all the proxies in a proxy network, thereby making the proxy network dysfunctional. The strategy is to compromise proxies and propagate along the proxy network topology. As shown in Figure 3.3, these attacks

allow attackers to propagate in the proxy network, increase the number of compromised proxies, and perhaps, eventually compromise all the proxies.

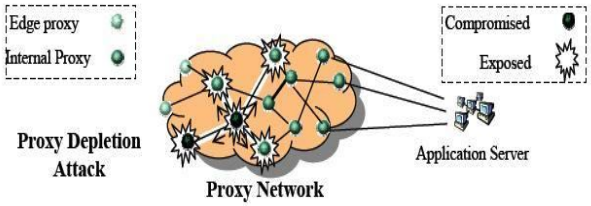


Figure3.3:-Proxy Depletion Attack

Both penetration attacks and proxy depletion attacks use the same mechanisms, host compromise attacks. As shown in Figure 3.4, host compromise attacks change the state of hosts and proxies. A successful host compromise attack changes an intact host to a compromised host. By compromising the host on which a proxy runs, an attacker can compromise the proxy. The neighbors of the compromised proxy then become exposed because attackers may learn their IP addresses from the compromised proxy.

Using host compromise attacks, we can construct both penetration attacks and proxy depletion attacks. In a penetration attack, attackers start from an edge proxy and use host compromise mechanisms to compromise the edge proxy. Once the proxy is compromised, all of its neighbor proxies become exposed. By compromising a sequence of exposed proxies along a path from the edge proxy to the application, attackers can penetrate the proxy network and eventually expose the application. On the other hand, in a proxy depletion attack, after compromising a proxy, attackers attack all the exposed neighbors, thereby propagating along the proxy network topology, increasing the number of compromised proxies.

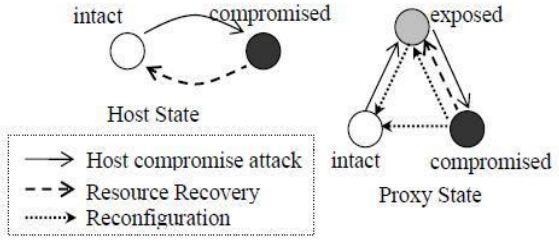


Figure 3.4:-System Components State Transitions

C) Defensive Mechanisms

The goal of defense is to reverse the negative impact of attacks on the system. Defenses can recover compromised hosts, making them intact, thereby increasing the population of intact hosts for proxy networks to use. Defenses can also turn compromised and exposed proxies into intact proxies, thereby reducing the population of compromised proxies and increasing the distance between exposed proxies and the application. We discuss two types of defense in the following section: resource recovery and proxy network reconfiguration.

Resource recovery mechanisms are defenses which address host compromise attacks. Examples of resource recovery include removal of infected software components, clean reload of system images with up-to-date security patches, revocation of suspected user accounts, and so on. Such resource recovery can eliminate attackers. control on compromised hosts and proxies, and also prevent future attacks using the same vulnerabilities of the hosts. We consider their use on all the hosts in the resource pool and trigger them using two policies: reactive recoveries and proactive resets. Reactive recoveries depend on intrusion or compromise detection, and are triggered when compromises are detected. In contrast, proactive resets happen periodically, regardless of the current state of the host.

They change the state of system components. At the host level (see Figure 3.4), resource recovery takes compromised hosts and returns them to the intact state. At the proxy level, resource recovery turns a compromised proxy into the exposed state by recovering the underlying host.

Proxy network reconfiguration is another type of defense. Reconfiguration can invalidate the location information acquired by attackers, and disrupt both penetration attacks and proxy depletion attacks. Examples include changing proxy network topology and proxy migration. We focus on .random proxy migration., where proxies can migrate from one host to another inside the resource pool, but the proxy network topology is unchanged. The migration mechanism is deployed on all the proxies in the proxy network, and every proxy (except edge proxies) periodically migrates randomly amongst hosts in the resource pool.

Proxy migration can change the state of proxies. As shown in Figure 3.4, proxy migration can turn an exposed or compromised proxy into an intact one, by moving the proxy to an intact host unknown to

attackers. Furthermore, this mechanism allows proxies to escape from exposed locations before they are compromised by attackers, thereby preventing the propagation of attacks and disrupting both penetration attacks and proxy depletion attacks.

3.2 Generality of the Generic Framework

Having defined a generic framework for proxy network-based DoS defense, we show how it captures several previously proposed proxy network schemes, including Secure Overlay Services (SOS) [18,19] and Internet Indirection Infrastructure (i3) [21, 22, 28]. Then, moving beyond specific examples, we discuss the space of proxy network-based DoS defense schemes captured by our framework

A) Secure Overlay Services

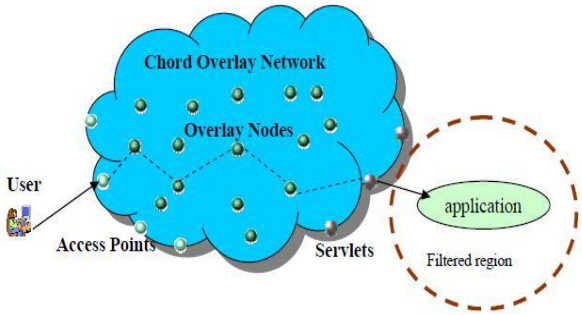


Figure 3.5:- Secure Overlay Services

As shown in Figure 3.5, Secure Overlay Services (SOS) is a proxy network scheme that uses the Chord overlay network [31] to mediate all traffic between users and applications and protect applications from DoS attacks. On one side of the Chord network, a set of overlay nodes (.access points.) publish their IP addresses and provide users access to the application. On the other side, a set of overlay nodes (.servlets.) connect to the application. Application traffic between users and applications is mediated through the Chord network via the access points and the servlets. Furthermore, filters are used around the application to enforce that only traffic from the servlets can reach the application, thereby preventing direct infrastructure-level DoS attacks on the application. Our generic framework captures the key properties of the SOS scheme as follows.

First, the key components of SOS system match those of our generic framework. The Chord network used by SOS can be represented using our generic proxy network with a Chord topology, the access points of SOS correspond to the edge proxies in our framework, and the .servlets. correspond to the proxies that directly connect to the application in our framework.

Second, the attack and defense processes described in our generic framework can apply to the SOS system. Regarding attacks, both penetration attacks and proxy depletion attacks described in our framework are key threats to the SOS system. Using penetration attacks, attackers can penetrate the Chord network and discover the IP addresses of the servlets. Once the servlets are exposed, attackers can easily defeat the SOS defense, because DoS attacks using packets spoofed with servlets. IP addresses can go through the filters, and reach the application. On the other hand, using proxy depletion attacks, attackers may compromise all the SOS nodes, thereby disabling the SOS system. Regarding defenses, both reactive and proactive resource recoveries described in our framework can directly apply to the SOS system. The SOS proposal does not include any proxy network reconfiguration mechanism.

B) Internet Indirection Infrastructure (i3)

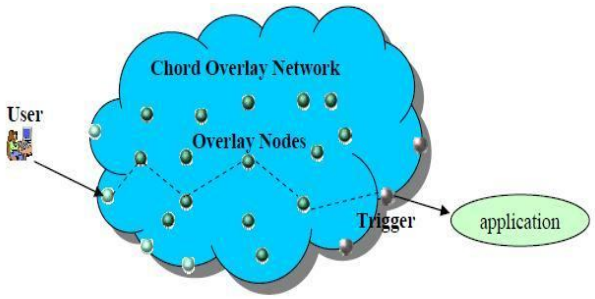


Figure 3.6:- Internet Indirect Infrastructure

Internet Indirection Infrastructure (i3) is another proxy network scheme that protects Internet services from DoS attacks. As shown in Figure 3.6, the i3 system uses a Chord overlay network to mediate all traffic between users and applications, protecting applications from DoS attacks. In the i3 system, the IP address of the application is hidden from users. On one side of the Chord network, a set of overlay nodes publish their IP addresses, providing users access to the Chord network. On the other side, an overlay node called “trigger” directly connects to the application and serves as a rendezvous point for the application. As such, i3 mediates application traffic through the Chord network and prevents direct infrastructure-level DoS attacks on the application. Our generic framework captures the key properties of the i3 scheme as follows.

First, the key components of the i3 system match those of our generic framework. The Chord network used by i3 can be represented using our generic proxy network with a Chord topology, the i3 nodes with published IP addresses correspond to the edge proxies in our framework, and the triggers correspond to the proxies that directly connect to the application in our framework.

Second, the attack and defense processes described in our generic framework can also apply to the i3 system. Regarding attacks, both penetration attacks and proxy depletion attacks described in our framework are key threats to the i3 system. Using penetration attacks, attackers can penetrate the Chord network and discover the IP addresses of the application, thereby exposing the application to direct DoS attacks. On the other hand, using proxy depletion attacks, attackers may compromise all the i3 nodes, thereby disabling the i3 system. Regarding defenses, both reactive and proactive resource recoveries described in our framework can apply to the i3 system directly. The i3 proposal does not include any proxy network reconfiguration mechanism.

C) Space of Proxy Networks

Besides the existing proxy network proposals, our generic framework admits DoS resistance schemes using a wide range of proxy networks, varying in topologies, depth and width, deployment schemes, and defensive mechanisms. For example, a proxy network may use a tree or a hypercube [33] as its topology instead of Chord. A proxy network may also employ defensive mechanisms such as proxy migration or dynamic change of proxy network topology.

Our generic framework provides a basis for a general exploration of the space of proxy networks. First, this framework allows study of fundamental capabilities and limitations of a large class of proxy network-based DoS defense schemes with results that bear on the entire class. Second, this framework also allows exploration of the design space of proxy networks, providing design guidelines for proxy network-based DoS defense.

4. Stochastic Model for System Component Dynamics

We model system state as a discrete-time stochastic process in which the state transitions of system components. Hosts and proxies are stochastic events. As such, we can quantify how attacks, defenses, correlated host vulnerabilities, and proxy network topology affect the system. Our stochastic model has two parts: host state transitions and proxy state transitions; Table 5-1 shows the parameters of the model. We first describe the model and then interpret the model in practical settings.

Table 4.1 Parameters of the Stochastic Model

Notation	Meaning
$\lambda 0$	Rate of host compromises based on new vulnerabilities
λv	Rate of host compromises based on known vulnerabilities
μs	Rate of proactive resets
μd	Speed of reactive recovery
μr	Rate of proxy migration

A) Host State Transitions

Attacks, resources recovery and correlated host vulnerabilities are the three main factors that affect the transitions of host states. We first describe how our model captures attacks and resources recovery when the host vulnerabilities are uncorrelated, we then describe how our model correlated host vulnerabilities.

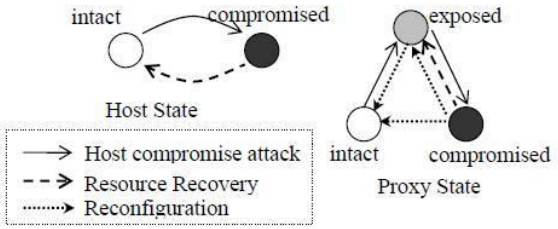


Figure 4.1 Host State transitions

The shaded area in Figurs shows the host state transitions when the host vulnerabilities are uncorrelated. Our model uses three parameters $\lambda 0$, μd, and μs to describe the speed of attacks, reactive resource recovery, and proactive resets, respectively. Within a discrete time step, attackers have a probability $\lambda 0$ to compromise an intact host by exploiting a vulnerability of the host. Meanwhile, reactive resource recovery has a probability μd to recover a compromised host by detecting and removing the infection, while proactive resets have a probability μs to recover a compromised host by proactively reloading the host with a clean system image.

Our model also captures correlated host vulnerabilities. We use "domains" to describe the correlated vulnerabilities among hosts (see Figure 4.2). Hosts are

grouped into domains. Within a domain, hosts use similar software with similar configurations, thereby sharing similar vulnerabilities. Across domains, hosts differ in software, configurations, and other attributes, thereby providing a model for uncorrelated vulnerabilities. A system with uncorrelated host vulnerabilities (see Figure 4.2.A) is an extreme case where each host is in its own domain. Another extreme case is one where all hosts are in the same domain (see Figure 5-2.B). In general, hosts in a system are grouped into multiple domains (see Figure 5-2.C), and the number of domains is a measure of host diversity in the system.

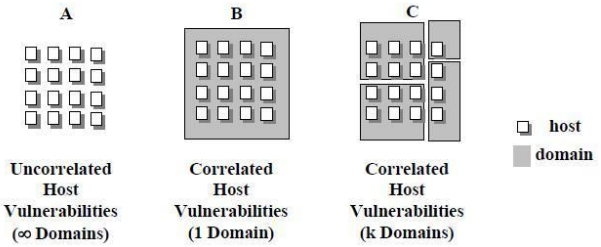


Figure 4.2 Domain Based Correlated Host Vulnerability Model

To model the impact of correlated host vulnerabilities, we introduce an intermediate host state "intactv" (an intact host with a known vulnerability) and one more parameter λv (see Figure 4.1). Here is the revised model. Within a discrete time step, with probability $\lambda 0$ attackers can compromise an intact host by exploiting a new vulnerability, changing the other intact hosts in the same domain to the "intactv" state. With probability λv attackers can compromise an "intactv" host by exploiting a known vulnerability. Meanwhile, with probability μs proactive resets can return a host from the "íntactv" state to the intact state, by removing the known vulnerabilities. With probability μd and μs, reactive recovery and proactive resets can return a compromised host to yje intact state respectively.

B) Proxy State Transition

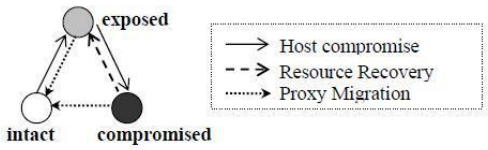


Figure 4.3 proxy State transition

A proxy's state depends on three factors: the state of the host where the proxy runs, the state of the neighbor an edge proxy. Based on the omised. the e μr to describe the proxy migration proc ctors of the system, including speed of attack, speed of defense, proxy network structure, and correlated host vulnerabilities. ing proxies, and whether or not the proxy is host state transition model described above, we can use the following rules to determine the state of a proxy under host compromise attacks.

- A proxy is compromised if and only if its host is.
- The neighbors of a compromised proxy are exposed, or compromised.
- All edge proxies are exposed or compromised.

Furthermore, proxy migration moves a proxy to a different host and changes proxy's state accordingly. We use migration rates, where proxies choose migration targets randomly and the migration overhead is small compared to the interval between migrations. More precisely, a proxy has probability μr to move to a different host within a discrete time step. After migration, the proxy's state is determined by the rules above.

5. Simulation Results: Correlated Vulnerabilities

We know that with proxy migration, proxy networks can resist penetration attacks effectively; the time to penetrate a proxy network increases exponentially with the proxy network's depth. However, analysis so far assumed uncorrelated host vulnerabilities. Typically, hosts share a range of correlated host vulnerabilities (e.g. exploitable bugs in the same software or operating systems, common configuration errors, same user accounts with same passwords), and compromising one host can increase the chance of compromising others significantly. In this section, we use a Monte-Carlo simulation to study systems in which hosts have correlated vulnerabilities. We first analyze how adding correlated host vulnerabilities affect the previous results, and what can be used to mitigate the negative impact of correlated host vulnerabilities. Then, based on these results, we study whether proxy networks can resist penetration attacks with correlated host vulnerabilities.

In the simulation, we choose λv to be close to 1, to represent highly correlated host vulnerabilities3; i.e. once attackers compromise a host, they can compromise any other host in the same domain with a high probability λv within the next time step (recall that hosts in a domain have highly correlated vulnerabilities, and hosts across domains are uncorrelated). $\lambda 0$ is set according to Table 4.2; other parameters are relative to $\lambda 0$, and can be easily inferred.

5.1 How Does Adding Correlated Host Vulnerabilities Affect Previous Results?

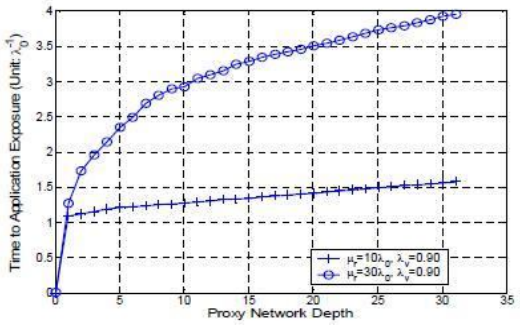


Figure 5.1 Impact of Proxy Network Depth with Correlated Host Vulnerabilities

To answer this question, we consider a system in which all hosts are in the same domain where the host vulnerabilities are highly correlated (λv=0.9) and the hosts do not use proactive resets to remove known vulnerabilities (μs=0). Figure 5-9 shows the time to application exposure as a function of proxy network depth with high proxy migration rates ($\mu r=10\lambda 0$ and $\mu r=30\lambda 0$ respectively) and instantaneous reactive resource recovery (μd=1, all hosts are recovered immediately after they are compromised). In Figure 5.1, the X-axis is proxy network depth, and the Y-axis is the time to application exposure.

Our simulation results show that correlated vulnerabilities have a major impact on a proxy network's resistance to penetration attacks. Recall that if host vulnerabilities are uncorrelated, the time to application exposure would increase exponentially with proxy network depth. However, both curves in Figure 5.1 stay flat, indicating that in a system with correlated host vulnerabilities, the time to application exposure does not increase much with proxy network depth, which means that the proxy network cannot resist penetration attacks effectively. Therefore, correlated host vulnerabilities can change a proxy network.s ability to resist penetration attacks qualitatively, thus dramatically reducing the effectiveness of defense.

5.2 How to Mitigate the Impact of Correlated Host Vulnerabilities?

Unless the negative impact of correlated host vulnerabilities can be mitigated, proxy networks cannot resist penetration attacks effectively. We consider two techniques for mitigation: proactive resets and host diversity. Proactive resets can remove known host vulnerabilities before they can be attacked, thereby mitigating the impact of correlated host vulnerabilities. Meanwhile, host diversity (recall that the degree of host diversity is the number of domains in the system) can reduce correlated host vulnerabilities because only hosts inside the same domain have correlated host vulnerabilities, and hosts in different domains are uncorrelated.

To study how proactive resets reduce the negative impact of correlated host vulnerabilities, we vary the proactive reset rate and study the penetration probability for proxy networks in a system of one domain (all the hosts have highly correlated vulnerabilities, λv=0.90). Specifically, for a range of proxy networks with varied depths, we measure the probability of penetrating them within 106 time steps under varied proactive reset rates. The results are shown in Figure 5.2. The X-axis is the depth of a proxy network, and the Y-axis is the probability of penetrating the proxy network within 106 time steps. Each curve corresponds to a proactive reset rate (μs). The case of uncorrelated host vulnerabilities is also shown for comparison; it displays a contrast to the uncorrelated case. A smaller difference indicates a better reduction of the negative impact of correlated host vulnerabilities. Figure 5.2 shows that even for high proactive reset rates, the impact of correlated host vulnerabilities is still prominent. This is because proactive resets are not guaranteed to happen before attacks, and known host vulnerabilities are not always removed before being attacked. Therefore proactive resets alone cannot contain the impact of correlated host vulnerabilities effectively.

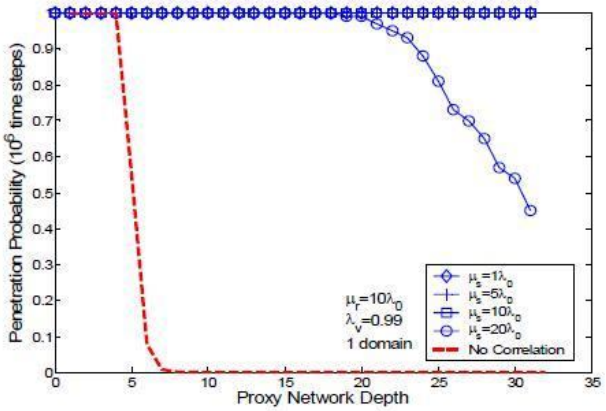


Figure:5.2 Penetration probability under Varied Proactive Reset Rates

We study whether adding host diversity into the system can reduce the negative impact of correlated host

vulnerabilities. In particular, at a fixed proactive reset rate (e.g. μs=10λ0) and a fixed proxy migration rate (e.g. μr=10λ0), we measure the probability of penetrating a proxy network in systems of varied degrees of host diversity. In each system, hosts are partitioned equally into k domains (k = 1, 2, 3, 4, 8), and proxies are placed randomly on the hosts. The results are shown in Figure 5-3. The X-axis is the depth of a proxy network, and the Y-axis is the probability of penetrating the proxy network within 106 time steps. Each curve corresponds to a certain degree of host diversity; the case of uncorrelated host vulnerabilities is also plotted for comparison, and shows a contrast to the uncorrelated case

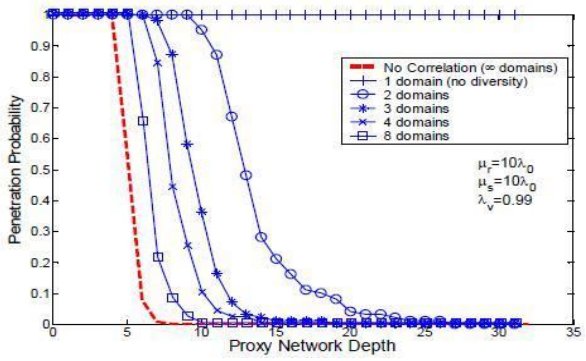


Figure 5.3 Penetration Probability under Varied Host Diversity

Figure 5-3 shows that adding even small degrees of host diversity into the system can reduce the impact of correlated host vulnerabilities significantly. In Figure 5.3, without host diversity, a proxy network of depth 32 can be penetrated within 106 time steps (with probability 1). In contrast, in a system with two domains, a proxy network of depth 25 cannot be penetrated within 106 time steps (penetration probability is close to zero); and in a system with three domains, a proxy network of depth 15 cannot be penetrated within 106 time steps.

5.3 Can Proxy Networks Resist Penetration Attacks with Correlated Vulnerabilities?

We have shown that host diversity and proactive resets can potentially counter the negative impact of correlated host vulnerabilities. However, as shown in Figure 5-3, a naïve scheme (proxies are randomly placed on hosts) is insufficient to remove the negative impact of correlated host vulnerabilities. The simple scheme has two main shortcomings.

First, placing proxies randomly allows neighboring proxies to run in the same domain, so their host vulnerabilities are correlated and they will fail together. A better approach is to place neighboring proxies on hosts in different domains, which will increase the effectiveness of the proxy network in slowing the attack progress.

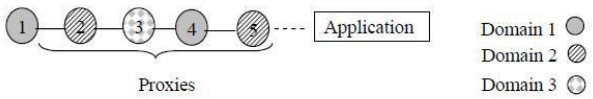


Figure 5.4 Interleaved Design for A proxy Chain

Second, allowing proxies to migrate to random hosts may help attackers, because a proxy may migrate to a host which has known vulnerabilities, allowing it to be compromised quickly, thereby improving the attack progress.

To address these issues, we develop an interleaved proxy network design where 1) proxy hosts are selected such that the distance is maximized between any pair of proxies in the same domain, and 2) proxy migrations are confined to hosts from the same domain. For example, as shown in Figure 5.4, we can place a chain of proxies to hosts of k domains using a round-robin order4.

To understand the effectiveness of the interleaved design in reducing the impact of correlated host vulnerabilities, we measure the probability of penetrating proxy networks using this design in systems with varied degrees of host diversity. The results for two proxy migration rates (μr=5λ0 and μr=10λ0) are shown in Figure 5.5. The X-axis is the depth of a proxy network, and the Y-axis is the probability of penetrating the proxy network within 106 time steps. Each curve corresponds to a certain degree of host diversity, and the case of uncorrelated host vulnerabilities is also plotted for comparison. In Figure 5.6, the curves for 4 and 8 domains closely follow the curve for the uncorrelated case. To verify this finding, we also study the system for longer time periods (10^7 and 10^8 time steps, see Figure 5.6), and observe the same phenomena . with 4 or more domains, the system behaves almost identically to one with uncorrelated vulnerabilities. This indicates that using a small degree of host diversity, e.g. 4 domains, our design can reduce the negative impact of correlated

host vulnerabilities significantly, and enable a proxy network to resist penetration attacks effectively.

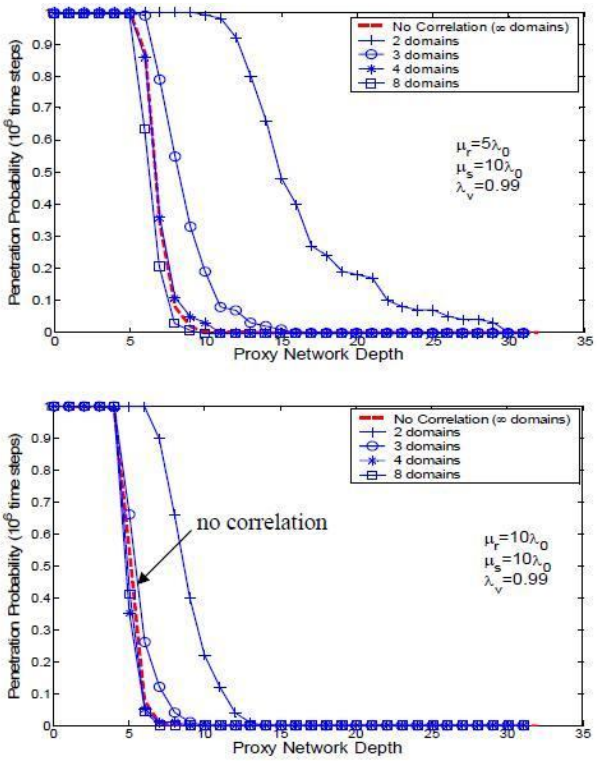


Figure 5.5 Effectiveness of Interleaved Design

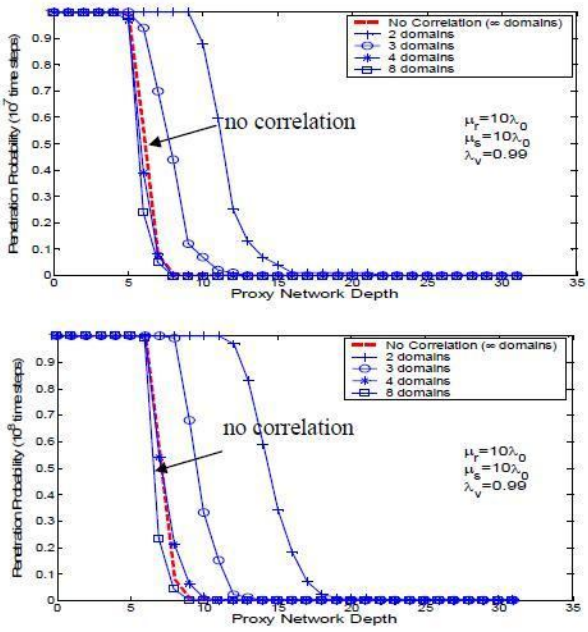


Figure 5.6 Effectiveness of Interleaved Design (Date point observed from 10^7 and 10^8 time steps)

Here is why a small degree of host diversity can be used for effective defense. In the interleaved design for a chain of proxies in a system of k domains (illustrated in Figure 5-4), between any two proxies (A and B) in the same domain there is a path of k-1 proxies in the different domains. After compromising proxy A, attackers must penetrate this path before they can attack proxy B. Since the penetration time grows exponentially with the path length (which is k-1), a small degree of host diversity (or the number of domains k) can provide a large penetration time5, allowing enough time for proactive resets to remove the known vulnerabilities on proxy B.s host (used for proxy A.s compromise) before they are attacked. Therefore, the interleaved design can reduce the impact of correlated host vulnerabilities significantly, thus enabling effective resistance to penetration attacks.

Future Work

The research described in this dissertation focused primarily on demonstrating the viability of the proxy network-based DoS defense as a system-level defense which can protect Internet service applications from infrastructure-level DoS attacks. While we believe that we were successful in meeting this goal, more advances can be made to improve the fidelity of the study, to cover a wider range of attack scenarios, to explore multiple dimensions of the design space (e.g. attack resistance, performance, and fault tolerance), and to investigate the use of proxy networks for defense against application-level DoS attacks.

The advantage of using a proxy network is the fact that it is inherently distributed and there is no fundamental resource limitation as opposed to a localized solution. For instance, a distributed filtering scheme implemented on a proxy network can potentially have a much larger capacity than any localized filters, and thus resist larger attacks. Future research may explore these potentials, and extend the proxy network-based DoS defense to a comprehensive architecture that can be used for defense against both infrastructure-level and application-level DoS attacks.

Summary

We develop a stochastic model based on the generic framework and use it to characterize the impact of attacks and defenses on the proxy network system. Based on this model, we combine analysis with Monte Carlo simulation to study proxy networks. resistance to penetration attacks. We show that,

• without reconfiguration mechanisms, a proxy network is vulnerable to penetration attacks,

• with proxy migration, a proxy network can resist penetration attacks effectively . the time to penetrate the proxy network grows exponentially with its depth, so that a moderate depth enables effective resistance to penetration attacks. Proxy network depth and proxy migration rates are the critical factors for achieving effectiveness.

• in many cases, correlated host vulnerabilities can make a proxy network vulnerable to penetration attacks.

• by exploiting the host (OS/software) diversity and intelligent proxy network construction, a proxy network can mitigate the negative impact of correlated host vulnerabilities and resist penetration attacks effectively.

References:-

[1} Dittrich, D., The DoS Project's "trinoo" distributed denial of service attack tool, 1999, University of Washington http://staff.washington.edu/dittrich/misc/trinoo.analysis.

[2]. Dittrich, D., et al., The "mstream" distributed denial of service attack tool, 2000. http://staff.washington.edu/dittrich/misc/mstream.analysis.txt.

[3]. Dittrich, D., The "Tribe Flood Network" distributed denial of service attack tool, 1999, University of Washington, http://staff.washington.edu/dittrich/misc/tfn.analysis.txt

[4].CERT, "Code Red" Worm Exploiting Buffer OverflowIn IIS Indexing ServiceDLL, 2001,Pittsburgh,PA, http://www.cert.org/incident_nes/IN2001-08.html.

[5].CERT, "Code Red II:" Another Worm Exploiting Buffer Overflow In IIS Indexing Service DLL, 2001, Pittsburgh, PA, http://www.cert.org/incident_ notes/IN-2001-09.html

[6].Moore, D., et al., The Spread of the Sapphire/Slammer Worm. 2003, CAIDA, UCSD, ICIR & LBNL, Silicon Defense, UC Berkeley

[7]. Hines, E.S., MyDoom.B Worm Analysis, 2004, Applied Watch Technologies, Inc., http://isc.sans.org/presentations/MyDoom_B_Analysis.pdf.

[8]. Williams, M., EBay, Amazon, Buy.com hit by attacks, 2000, http://www.nwfusion.com/news/2000/0209attack.html.

[9]. Fonseca, B., Yahoo outage raises Web concerns, 2000, http://www.nwfusion.com/news/2000/0209yahoo2.html.

[10]. CSI/FBI, Cyber Attacks Continue, but Financial Losses are Down 2003, http://www.gocsi.com/press/20030528.jhtml?requestid=335314.

[11]. Moore, D., G.M. Voelker, and S. Savage. Inferring Internet Denial-of-Service Activity. in proceedings of the 2001 USENIX Security Symposium. 2001.

[12]. Ferguson, P. and D. Senie, Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing.The Internet Society, 1998.

[13]. Cisco, Defining Strategies to Protect Against TCP SYN Denial of Service Attacks, http://cio.cisco.com/warp/public/707/4.html.

[14]. Cisco, Using CAR During DOS Attacks, http://www.cisco.com/warp/public/63/car_ratelimit_icmp.html.

[15]. Song, D.X. and A. Perrig. Advanced and authenticated marking schemes for IP traceback. in 20th Annual Joint Conference Of the IEEE Computer and Communications Societies. 2001.Anchorage, AK, United States: Proceedings – IEEE INFOCOM.v 2 2001.

[16]. Snoeren, A.C., et al. Hash-based IP traceback. in ACM Special Interest Group on Data Communications (SIGCOMM).2001. San Diego, CA, United States: Computer Communication Review. v 31 n 4 2001.

[17]. Savage, S., et al., Practical network support for IP traceback.Computer Communication Review, 2000. 30(4): p. 295-306.

[18]. Stavrou, A., et al., WebSOS: An Overlay-based System For Protecting Web Servers From Denial of Service Attacks. ElsevierJournal ofComputer Networks, special issue on Web and Network Security, 2005.

[19]. Keromytis, A.D., V. Misra, and D. Rubenstein. SOS:Secure Overlay Services. in ACM Special Interest Groupon Data Communications (SIGCOMM). 2002. Pittsburgh, PA:ACM.

[20] Andersen, D.G. Mayday: Distributed Filtering for Internet Services. in 4th Usenix Symposium on Internet Technologies and Systems. 2003. Seattle, Washington.

[21]. Adkins, D., et al., Towards a More Functional and Secure Network Infrastructure. 2003, Computer Science Division, UC Berkeley: Berkeley

[22]. Adkins, D., et al. Taming IP Packet Flooding Attacks. in HotNets-II. 2003.

[23]. Keromytis, A.D., V. Misra, and D. Rubenstein. Using Overlays to Improve Network Security. in the ITCom Conference, special track on Scalability and Traffic Control in IP Ne 2002.

[24]. Keromytis, A., V. Misra, and D. Rubenstein, SOS:An Architecture For Mitigating DDoS Attacks. IEEE Journal on Selected Areas of Communications (JSAC), 2004. 21(1): p. 176-188.

[25]. Ioannidis, S., et al. Implementing a Distributed Firewall. in the 7th ACM International Conference on Computer and Communications Security (CCS). 2000.

[26]. Xuan, D., S. Chellappan, and X. Wang. Analyzing the Secure Overlay Services Architecture under Intelligent DDoS Attacks. in 24th International Conference on Distributed Computing Systems (ICDCS'04). 2004.

[27]. Lakshminarayanan, K., et al. Towards a Secure Indirection Infrastructure. in ACM Symposium on Principles of Distributed Computing. 2004.

[28]. Stoica, I., et al. Internet Indirection Infrastructure. in ACM Special Interest Group on Data Communications

(SIGCOMM). 2002.

[29]. Akamai, Akamai Technology Overview, http://www.akamai.com/en/html/technology /overview.html.

[30]. Liu, X. and A.A. Chien. Realistic Large-Scale Online Network Simulation. in SuperComputing'04. 2004. Pittsburgh, PA.

[31]. Stoica, I., et al. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. in ACM Special Interest Group on Data Communications (SIGCOMM). 2001.

[32]. Ratnasamy, S., et al. A Scalable Content-Addressable Network. in ACM Special Interest Group on Data Communications (SIGCOMM). 2001.

[33]. Leighton, F.T., Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes. 1991: Morgan Kaufmann Pub.

[34]. Liu, X., H. Xia, and A.A. Chien, Validating and Scaling the MicroGrid: A Scientific Instrument for Grid Dynamics. Journal of Grid Computing, 2003.

[35]. Liu, X. and A. Chien. Traffic-based Load Balance for Scalable Network Emulation. in SuperComputing 2003. Noverber 2003. Phoenix, Arizona: the Proceedings of T he ACM Conference on High Performance Computing and Networking.

.