

# Understanding When Location-Hiding Using Overlay Networks is Feasible

Ju Wang and Andrew A. Chien

Department of Computer Science and Engineering, UC San Diego

## *Abstract*

*Overlay networks (proxy networks) have been used as a communication infrastructure to allow applications to communicate with users without revealing their IP addresses. Such proxy networks are used to enhance application security; including protecting applications from direct attacks and infrastructure Denial-of-Service attacks. However, the conditions under which such approaches can hide application location are not well understood. To shed light on this question, we develop a formal framework for proxy network approach to location-hiding which encompasses most of the proposed approaches. It is used to characterize how attacks, defenses, and correlated host vulnerabilities affect the feasibility of location-hiding.*

*We find that existing approaches employing static structures (e.g. SOS and I3) cannot hide application location because attackers gain information monotonically and quickly penetrate the proxy network. However, adding defenses, such as proxy network reconfiguration or migration, which invalidate the information attackers have, makes location-hiding feasible against penetration attacks. Proxy-network depth and reconfiguration rates are critical factors for effectiveness. Furthermore, correlated vulnerabilities in many cases jeopardize location-hiding; however, by exploiting host diversity and intelligent proxy-network construction, the negative impact of correlation can be mitigated and location-hiding can be achieved. These results provide deeper understanding of the location-hiding problem and guidelines for proxy-network design.*

## I. INTRODUCTION

Overlay networks have been proposed as a means to provide applications with new security capabilities. In particular, overlay networks are used as proxies which mediate the communication

between applications and their users without revealing application IP addresses. This capability to allow communication without revealing IP addresses is also known as location-hiding or application hiding, and the essence of it is indirect communication. This capability can support anonymous communication, protect applications and hosts from direct attacks, and protect the supporting physical infrastructure of an application from Denial-of-Service (DoS) attacks. For example, many researchers [1-4] exploit this location-hiding capability to protect Internet applications from DoS attacks on an application’s supporting physical infrastructure<sup>1</sup>.

However, fundamental questions about such proxy networks remain unclear, especially in the presence of intelligent attackers capable of compromising proxies: Can proxy networks provide anonymous communication and achieve location-hiding for applications via indirection? If so, under what circumstances can they stably withstand attacks? How long will it take attackers to penetrate a proxy network and reveal application location?

To shed light on these questions, we develop a generic framework for proxy network approaches to location-hiding, which encompasses most of the proposed approaches. We also develop a stochastic model to characterize the dynamic behavior of the system — exploring in particular how attacks, defense mechanisms, and correlated host vulnerabilities affect the ability to resist attacks. Based on this framework and model, we combine analysis and Monte Carlo simulation techniques to analyze the behavior of proxy network systems, and characterize when location-hiding is feasible and when it is not.

We focus on penetration attacks which exploit the connection structure of proxy networks, and use directed penetration in an attempt to reveal the application’s hidden location. Such attacks focus on elements that are present in all proxy network approaches. We do not focus on the attacks

---

<sup>1</sup> SOS[4] and Mayday[3] hide “secret servlets” instead of applications. See Section IV for detail.

which exploit weaknesses of the protocols used in specific instances of proxy networks (e.g. SOS). Such attacks often differ dramatically based on details of the particular scheme or even implementation, and therefore do not apply to proxy network approaches in general. We also do not consider attacks which compromise the entire resource pool, as in such cases, location-hiding is moot.

The specific research contributions of the paper include:

- design a generic framework and an analytic model for proxy network-based location-hiding,
- characterize fundamental properties of proxy network-based location-hiding, showing that existing approaches employing static structures [1-4] are vulnerable to penetration attacks.
- prove that proxy network-based location-hiding can be successful if it includes proactive defenses such as proxy network reconfiguration and proxy migration. The proxy network depth<sup>2</sup> and reconfiguration rates are critical for effective resistance.
- show that in many cases correlated host vulnerabilities make proxy network-based location-hiding infeasible.
- demonstrate that by exploiting the limited host (OS/software) diversity and intelligent proxy network construction, the negative impact of correlated vulnerabilities can be mitigated, and location-hiding can be achieved.

These results provide both deeper understanding of the location-hiding problem and guidelines for proxy network design. The generic framework provides a foundation to understand proxy networks' capability of location-hiding, to formally analyze the behavior of such systems, to rigorously reason about how proxy networks should be designed, and to serve as stepping stones for future studies based on more complex and realistic models.

---

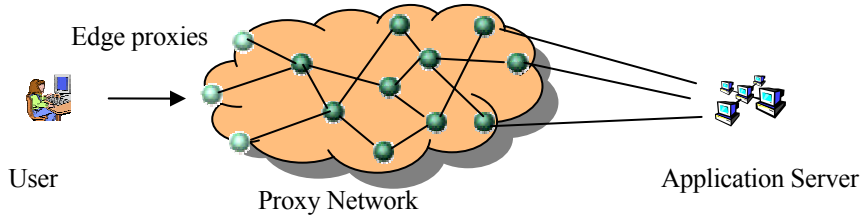
<sup>2</sup> Depth of a proxy network is the distance (overlay hop count) from its edge to the hidden node. It is formally defined in Section II.

The remainder of the paper is structured as follows. Section II describes the generic framework. Section III analyzes the location-hiding problem. Then, Section IV discusses a few additional important issues in location-hiding, and relates our work to the other studies. Finally, Section V concludes with a brief summary and a description of future directions.

## II. GENERIC FRAMEWORK

We define a generic proxy network for location-hiding, which encompasses a wide range of proposed approaches [1-4]. Then, we introduce a formal framework, defining the key system components, state transitions, and a stochastic model which characterizes dynamic system behavior.

### A. *Proxy Network Approach for Location-Hiding*



**Figure 1 Proxy Network-based Location-Hiding**

Figure 1 shows a generic proxy network-based location-hiding scheme encompassing most of the proposed approaches [1-4]. The essence of the proxy network approach is to mediate the communication between users and the application without revealing the application’s IP address. A proxy network is an overlay network that serves as an application mediator to support communication between an application and its users. As shown in Figure 1, the application is hidden behind the proxy network which mediates the application messages between the application and its users. On one side of the proxy network, a set of 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. In this way, users access the edge proxies to

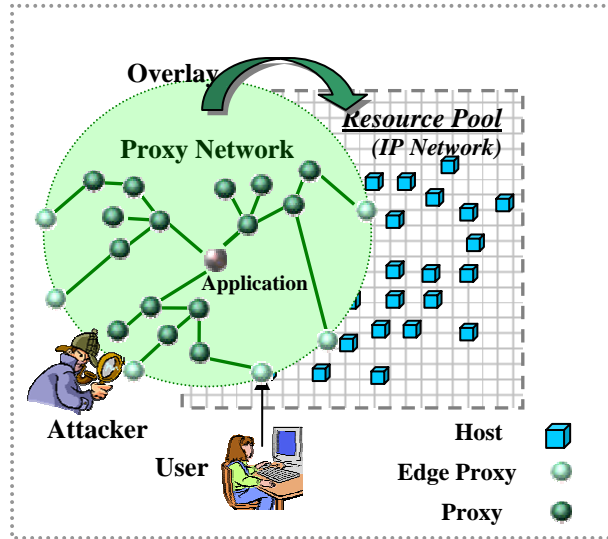
communicate with the application via the proxy network, and the IP address of the application is kept secret.

We assume that the proxy network operates in a large resource pool of tens of thousands or even millions of Internet hosts. Given the enormous size of the resource pool, it is practically impossible for an adversary to correctly guess the IP address of a certain proxy node. The feasibility of such large resource pools is demonstrated by existing infrastructure of large-scale distributed systems, such as content delivery networks and peer-to-peer systems. For example, Akamai network has over 15,000 servers deployed in over 1,200 networks across 65 countries [5]; peer-to-peer systems, such as Skype [6] and BitTorrent [7], typically have millions of hosts in their system. Furthermore, the number of hosts in the Internet is increasing rapidly, thus the size of the resource pools that can be built will increase according in the near future. Therefore, it is feasible to have a large resource pool of millions of hosts.

Furthermore, we assume that adversaries cannot identify proxy nodes by probing the resource pool. In practice, we can implement a proxy network which at any time uses a small portion of the enormous resource pool (as previously described). Moreover, the proxy program is implemented in such a way (e.g. using firewalls) that its presence cannot be easily discovered by probing remotely. In this way, it is difficult for an adversary to identify proxy nodes by probing the resource pool.

## ***B. System Components***

Figure 2 shows the five key components of the system: the application and users, proxies and hosts, and attackers. In this subsection, we define these components and their key properties.



**Figure 2 Generic Framework for Proxy Networks**

**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, the IP address of the application is hidden<sup>3</sup>; the application has connections with the proxy network, through which the application communicates with its users.

**B) User**

A user is the principal that uses the application client software to access the application interactively, 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, users are outside the proxy network and access the application via edge proxies (defined below) and through the proxy network.

---

<sup>3</sup> Some proposals (SOS and Mayday) hide “secret servlets” instead of applications. But our framework still applies. See Section IV for detail.

### **C) Host**

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 forms a resource pool for the proxy network.

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 2, 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.

There are two important properties of a proxy network: *topology* and *depth*.

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.

## **E) Attacker**

Attackers are adversaries to the proxy network scheme. Their goal is to discover the hidden application's location (*application exposure*). Similar to users, attackers are also outside the proxy network, and have all the information publicly available to users. Meanwhile, attackers are assumed to have capabilities enabling them to collect more information. In particular, in this paper, we study attackers who are capable of compromising hosts and proxies. Furthermore, we distinguish attacks that exploit the adjacency structure of a proxy network and the ones that try to compromise the entire resource pool (e.g. worm attacks). They are both threats to the proxy network approach, but have very different properties, and should be studied separately. In this paper, we focus on the former one which exploits the adjacency structure of proxy networks.

## **C. System Dynamics**

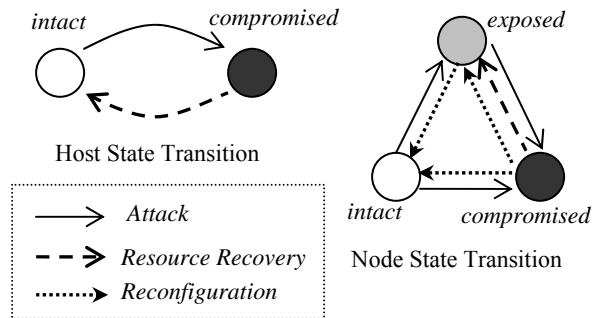
System dynamics describes the changes in system state which result from attacks and defenses. By studying the system dynamics of a proxy network under various attack and defense scenarios, we can understand when the proxy network can provide stable defense against penetration attacks. We first introduce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

### **A) System State**

The system state consists of host state and overlay node state. A host has two states: *compromised* and *intact*. It is *compromised* when attackers have control over it and any (adjacency) information stored there may be revealed to attackers. A host not compromised is *intact*. A node

(proxy or application) has three states: *intact*, *exposed*, and *compromised*<sup>4</sup>. It is *exposed* if attackers know its location; in this case it is subject to future attacks. It is *compromised* if it runs on a compromised host. It is *intact* if it is neither exposed nor compromised.

All nodes adjacent to a compromised node are exposed, since attackers can get the location information from the compromised node. Edge proxies are always exposed because their location is published for users to access. As stated before, using the published location information, attackers can employ host compromise mechanisms to exploit the structure of a proxy network, and discover location information about proxies and the application.

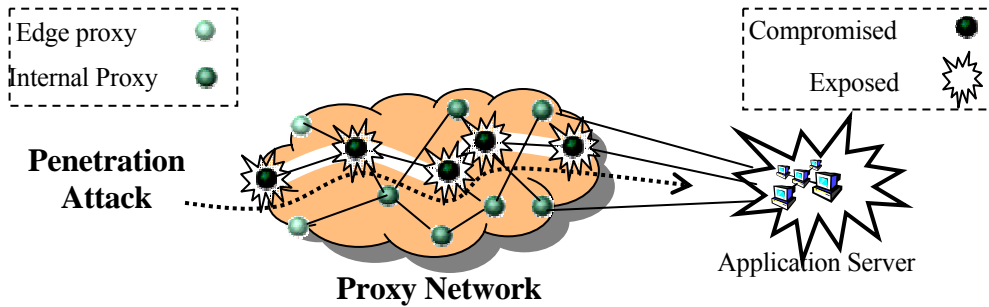


**Figure 3 System State Transition Illustration**

## B) Attacks

Our generic framework captures a range of attacks, among which we study penetration 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 4, 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.

<sup>4</sup>Terms “intact” and “compromised” are overloaded for overlay nodes and hosts.



**Figure 4 Penetration Attacks**

Penetration attacks are implemented using host compromise attacks, which encompass a wide range of attacks that can reveal the information stored on the victim host, or allow attackers to eavesdrop on network traffic<sup>5</sup>. Such attacks can be done remotely by exploiting vulnerabilities on a host. Other forms of attacks not captured in our framework are considered in Section V. As shown in Figure 3, 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 penetration attacks as follows. 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.

In addition, hosts may have *correlated vulnerabilities*. When attackers successfully discover and exploit a new vulnerability, all the hosts with similar vulnerabilities can more easily (quickly) be compromised.

<sup>5</sup> For example, if attackers can eavesdrop on a host, that host is considered compromised. For succinctness, we do not distinguish such attacks from host compromises.

## C) Defensive Mechanisms

The goal of defensive mechanisms is to reverse the negative impact of attacks on the system. There are two defensive mechanisms: resource recovery and proxy network reconfiguration.

*Resource Recovery* takes compromised hosts and returns them to an intact state. 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. Our model of resource recovery patches machines for all known vulnerabilities, including the one resulting in the last compromise. There are two triggering policies: reactive recoveries and proactive resets. Reactive recoveries depend on detection, and happen after compromises are detected. Proactive resets do not depend on detection, and reset hosts into the intact state regardless of their current state.

*Proxy Network Reconfiguration* changes the location of proxies and (or) the structure of the proxy network. These techniques invalidate the location information acquired by attackers (turning exposed nodes to intact ones) and disrupt attackers' penetration. We consider a simple form of reconfiguration – proxy migration: where proxies can migrate from one host to another inside the resource pool, but the proxy network topology is unchanged. Proxy migration allows exposed proxies to move to locations unknown to attackers, and therefore become intact<sup>6</sup>. In this paper, we study the case where proxies choose hosts randomly from the resource pool during migration, and the overhead of migration is small compared to interval between migrations. More complex migration schemes involving intelligent host selection are future work. Furthermore, implementation of proxy migration may affect the system behavior. Here we assume that when a

---

<sup>6</sup> In fact, there are three scenarios: if the new host the proxy moves to is compromised, the proxy becomes compromised; if any of this proxy's neighbors are compromised, then this proxy is exposed; if neither of the above happens, then this proxy becomes intact.

proxy migrates from a compromised host to an intact host, it does not carry the compromised components with it. Since we can move a proxy by stopping its current execution and starting a new instance on the destination host, it is straightforward to meet this assumption requirement. Section IV discusses the implementation issues of proxy migration in greater details.

**D. Stochastic Model**

We model dynamic changes in system state as a discrete-time stochastic process. Attacks and defenses drive this process. Correlated host vulnerabilities and proxy network topologies also affect it.

Correlated host vulnerabilities are prevalent in the Internet. We use a *domain*-based correlation model to characterize representative scenarios in practice, where hosts are grouped into “*domains*”. Within a domain, hosts use similar software (including operating systems), hardware, and configurations, so their vulnerabilities are correlated. Across domains, hosts differ in software, configuration, and other attributes, providing a model for uncorrelated vulnerabilities. The number of domains is a measure of *host diversity* in the system.

Notation	Meaning
$\lambda_0$	Rate of host compromises based on new host vulnerabilities
$\lambda_v$	Rate of host compromises based on known host vulnerabilities
$\mu_s$	Rate of proactive reset
$\rho$	True positive ratio of reactive recovery
$\mu_d$	Speed of reactive recovery
$\mu_r$	Rate of proxy migration

**Table II-1 Stochastic Model**

## A) Host State Transition

We view attackers as a single entity – a group of colluding attackers with a certain technological and resource capability. Key parameters of the model are shown in Table II-1,  $\lambda_0$  is the rate at which new vulnerabilities in a particular domain are discovered and exploited.  $\lambda_v$  is the rate of host compromises using known vulnerabilities. It also describes the level of correlation inside a domain: once a host is compromised,  $\lambda_v$  shows how fast attackers can compromise other hosts in the same domain (with similar configuration). From the perspective of a discrete-time stochastic process,  $\lambda_0$  and  $\lambda_v$  are the probabilities of turning intact hosts to the compromise state in a single time step.

Furthermore,  $\lambda_0$  is typically significantly smaller than  $\lambda_v$  ( $\lambda_0 \ll \lambda_v$ ). Studies on computer vulnerabilities and attack incidents [8, 9] showed that discovery and exploitation of new vulnerabilities is typically hard, requiring a significant amount of time and expertise on the victim system. Compromising a host with a known bug is fairly easy after the initial “research stage”.

Both reactive recovery and proactive reset change compromised hosts to intact, and remove known vulnerabilities. We use two parameters to characterize (detection-triggered) reactive recovery, since not all compromises are detectable:  $\rho$  is the true positive ratio of the detectors (the fraction of all compromises which are detectable), and  $\mu_d$  is speed of reactive recovery, or the probability of recovering a detectable compromise within a time step.  $\mu_s$  is the rate of proactive reset – a host has the probability  $\mu_s$  to be reset proactively within a single time step.

## B) Proxy State Transition

A proxy's state depends on the state of its host. A proxy is compromised if and only if its host is compromised. At any moment, all neighbors of a compromised proxy are exposed. Furthermore, all edge proxies are always exposed.

We use one parameter, migration rate, to describe the migration process, where proxies migrate randomly among hosts, and the migration overhead is negligible compared to the interval between migrations. Proxies migrate at rate  $\mu_r$  – in a time step, a proxy has the probability  $\mu_r$  to move to a different host. This proxy becomes intact if its new host is intact and none of its neighbors are compromised.

### *E. Discussion*

Since little is understood about location-hiding, we choose a simple model to keep the problem tractable and to build intuition. The model, while simple, captures all the key factors of the system, including the speed of attack, the speed of defenses, proxy network structure, and correlated host vulnerabilities. These factors together decide how the system state changes over time. For example, we use one rate to describe the discoveries of new vulnerabilities (a Poisson rate in the stochastic sense). Previous research [10, 11] on vulnerabilities in large computer systems shows that Poisson process can approximate the discovery of vulnerabilities on large systems, justifying this simple model.

To get an idea of the practical values of these model parameters, we collected numbers from real systems.

- $\lambda_0$  is the rate of new vulnerability discovery. Microsoft security bulletin [12] catalogues the critical vulnerabilities (remotely exploitable) of Windows XP Professional and Windows 2K

Server. Table II-2 shows the number of new vulnerabilities discovered for each period. There are about 20 new vulnerabilities discovered each year. On average, a new vulnerability is discovered every two to three weeks.

Year	2001	2002	2003	2004
WinXp Pro	5	20	19	18
Win2K Server	28	24	19	18

**Table II-2 Windows Vulnerability Statistics**

- $\lambda_v$  is the rate of host compromises using known vulnerabilities. An extreme case is that hosts are compromised using the exact same bug (as in worm propagation), where it only takes minutes or even less to compromise a host.
- $\rho$  is the true positive ratio of intrusion detection systems (IDS), and  $\mu_d$  is the speed of reactive recovery. Previous research on IDS [13, 14] indicates that modern IDS can achieve  $\rho$  over 0.80 and achieve real time detection.  $\mu_d$  is primarily determined by how fast a detected compromise can be removed.
- $\mu_s$  indicates how frequent a host is cleanly reset and patched. Note that we normally compare it to  $\lambda_0$  in the analysis, e.g. from Table II-2 we may say that  $\mu_s=3\lambda_0$  corresponds to a weekly reset.
- $\mu_r$  is the migration rate. Since proxies do not contain persistent data, it is straightforward to implement proxy migration. Our prototype implementation<sup>7</sup> on a local area network has a sub-second migration overhead. This suggests that current technology can support daily or even hourly proxy migration rates, i.e. 10x~100x higher than  $\lambda_0$ .

---

<sup>7</sup> Since the implementation detail of proxy migration has little impact on our analysis, we omit it for succinctness. A straightforward implementation can simply stop the current running proxy program and start a new instance at the new place. A simple protocol can be used to transfer the connection information to the new instance. Before migration, a proxy can flush all the application messages in its buffer to its neighbors; therefore no application messages are dropped.

### III. LOCATION-HIDING

We study the location-hiding problem in this section: How long will it take attackers to expose an application? Can proxy networks hide location? How do the properties of defense affect a proxy network’s capability of location-hiding? How do correlated host vulnerabilities affect location-hiding?

Since the system is difficult to analyze when correlated host vulnerabilities are considered, our study has two parts. First, we study the case of uncorrelated host vulnerabilities ( $\lambda_0=\lambda_v$ ), and provide two theorems to characterize the dynamic system behavior. We show that, with appropriate defense, location-hiding is feasible. Second, we use a Monte Carlo simulation to study the case with correlated host vulnerabilities. In particular, we show that, in many cases, correlated host vulnerabilities can jeopardize location-hiding. However, by exploiting host diversity and intelligent proxy network construction, we can build a proxy network which can achieve location-hiding, and behaves similarly to the uncorrelated case described by the Theorems. Combining both cases, we address the key questions.

#### A. *Analytical Study: Uncorrelated Vulnerabilities*

In this section we study analytically the case of uncorrelated vulnerabilities. This analysis provides a baseline understanding of the location-hiding problem, and a basis for a more general analysis. We focus on two basic questions: Can proxy networks hide location at all? How do the properties of defense affect location-hiding?

First, we present two theorems which quantify the expected time for attackers to penetrate a proxy network and expose the application, when host vulnerability is uncorrelated. Proofs of these theorems are provided in Appendix I and Appendix II. Equipped with these theorems, we study the location-hiding problem.

**Theorem I**

*Without proxy network reconfiguration, the expected time to application exposure  $T \leq dT_\lambda$  where  $T_\lambda$  is the expected time to compromise a host and  $d$  is the proxy network depth.*

**Theorem II**

*Consider a proxy network with random proxy migration rate  $\mu_r$ , which is sufficiently high ( $\mu_r \geq 2\lambda$ ), the expected time to application exposure  $T$  grows exponentially with the proxy network depth  $d$ ; specifically  $\Theta\left(\left(\frac{\mu_r}{2\lambda}\right)^{d-2}\right)T_\lambda \leq T \leq \Theta\left(\left(\frac{\mu_r}{\lambda}\right)^{d-1}\right)T_\lambda$ , where  $\lambda$  is the host compromise rate<sup>8</sup>, and  $T_\lambda = \lambda^{-1}$  is the expected time to compromise a host.*

**A) Can Proxy Networks Hide Application Location?**

Theorem I shows that, without proxy network reconfiguration, the time to penetrate a proxy network grows linearly with its depth (recall that depth of a proxy network is the distance between the edge proxies and the application) – an attacker can expose the application as easily as compromising a small number of hosts. In other words, without reconfiguration, a proxy network is vulnerable to penetration attacks, and cannot achieve location-hiding. Intuitively, without reconfiguration, attackers gain information monotonically – once a proxy becomes exposed it remains so. Therefore, to expose the application, attackers only need to compromise all the proxies on a path from an edge proxy to the application.

On the other hand, Theorem II shows that, with proxy migration (reconfiguration) at an appropriate rate, the time to expose an application grows exponentially with proxy network depth. Thus, small increases in proxy network depth (and therefore small costs in application overhead) can improve location-hiding significantly. Consequently, proxy networks of a modest depth and reconfiguration can effectively resist penetration attacks. For example, using the numbers in Table

---

<sup>8</sup> We use  $\lambda$  instead of  $\lambda_0$  here to distinguish this uncorrelated case from the general case with correlated vulnerabilities.

II-2, we assume that attackers take two weeks to compromise a host. (Note that host compromises are independent here, since uncorrelated host vulnerabilities are assumed.) If proxies migrate once per day ( $\mu_r \approx 10\lambda$ ), then penetrating a proxy network of depth four takes about fifty years, a depth six takes about five thousand years, eliminating this type of attack as a practical concern.

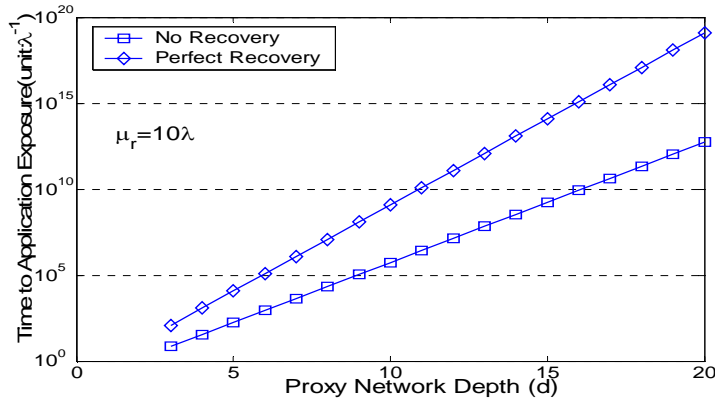
In summary, without reconfiguration, it is infeasible for a proxy network to achieve location-hiding. With proxy migration (reconfiguration), location-hiding is not only feasible, it has excellent scaling properties.

## **B) How Do Defenses Affect Location-Hiding?**

There are three key defense properties: proxy network depth, proxy migration rate, and resource recovery. To understand the impact of resource recovery schemes, we plot two cases: no recovery and perfect recovery. With “no recovery”, compromised hosts are never recovered (this case assumes an infinite resource pool). With “perfect recovery”, a host is recovered immediately after it is compromised; if a proxy runs on this host, then the proxy and its neighbors become exposed. These two cases provide an envelope for general cases with any resource recovery schemes.

### *Impact of Proxy Network Depth*

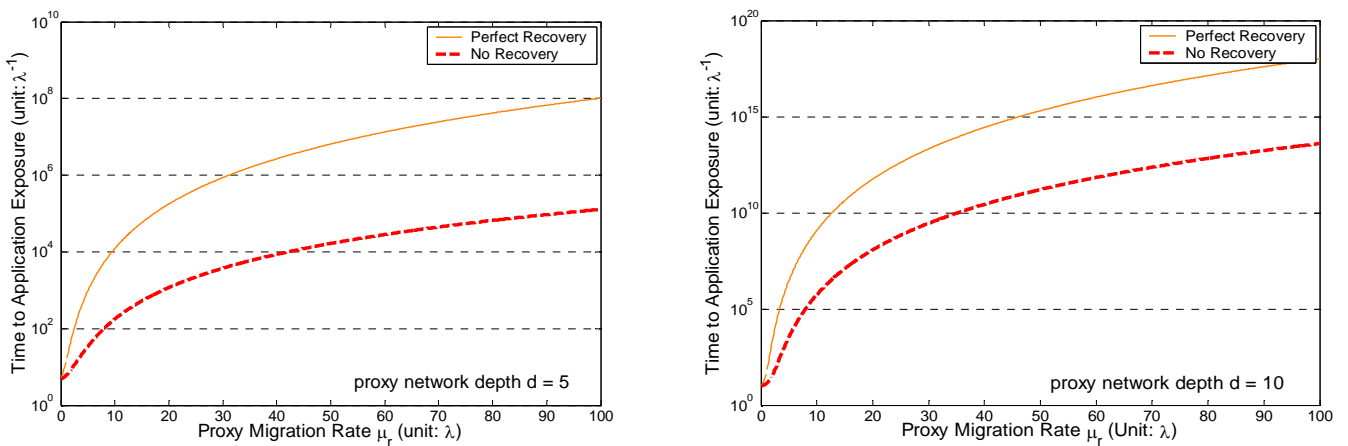
Figure 5 shows time to application exposure as a function of proxy network depth when proxy migration rate is sufficiently fast ( $\mu_r \geq 2\lambda$  according to Theorem II). Time to application exposure increases exponentially with proxy network depth (note the log scale). This indicates that proxy networks can be an effective barrier to penetration attacks, and achieve location-hiding.



**Figure 5 Impact of Proxy Net Depth (fast migration)**

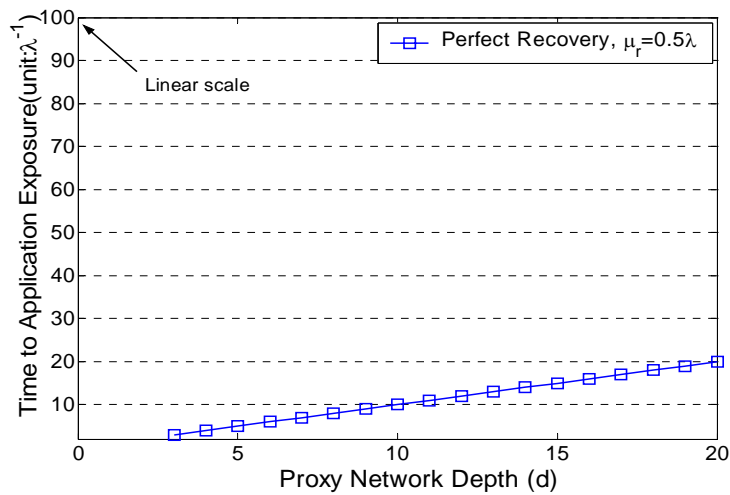
Impact of Proxy Migration

Figure 6 shows how proxy migration affects the expected time to application exposure (for cases of proxy network depth  $d$  of 5 and 10 respectively). The clear trend is that increased migration rate increases the expected time to application exposure significantly (note the log scale). Time to application exposure increases at a polynomial rate with  $d$  as the exponent. For example, when proxy network depth  $d$  is 10, doubling the migration rate can increase the time to application exposure by 1000 times.



**Figure 6 Impact of Proxy Migration**

More importantly, proxy migration rate has a critical impact: it needs to be sufficiently fast ( $\mu_r > 2\lambda$ ) to guarantee effective defense against attacks. Figure 7 shows how time to application exposure changes with proxy network depth when proxy migration is slow. In this case, time to application exposure no longer grows exponentially with the proxy network depth (note the Y-axis is not log scale), and location-hiding cannot be achieved. Comparing Figure 5 and Figure 7 makes it clear that proxy migration rate is a critical factor for location-hiding; proxy migration rate can change the effectiveness of location-hiding qualitatively.



**Figure 7 Impact of Proxy Net Depth (slow migration)**

Impact of Resource Recovery

Figure 5 and Figure 6 show that resource recovery schemes have moderate impact on location-hiding. Adjusting proxy migration rate and the proxy network depth can compensate for poor resource recovery, as long as there are sufficient intact hosts in the resource pool. However, this does not imply that good resource recovery schemes are unnecessary. Good recovery schemes can quantitatively improve location-hiding (both figures show that a perfect recovery scheme can increase the time to application exposure by several magnitudes over the case without recovery). Furthermore, resource recovery schemes can recover compromised hosts, thereby sustaining intact

hosts in the resource pool; they also help to overcome the negative impact of correlated host vulnerabilities as discussed in Section B.

### ***B. Simulation Study: Correlated Host Vulnerabilities***

The previous section shows that, with proxy migration, proxy networks can effectively hide application location – time to application exposure increases exponentially with proxy network depth. However, this is under the assumption of uncorrelated host vulnerabilities. In practice, host vulnerabilities are correlated. Compromising a host may significantly increase the chance (speed) of compromising others which share similar vulnerabilities. In this section, we use a Monte-Carlo simulation to study a more realistic case where hosts have correlated vulnerabilities.

First, we analyze how correlation affects the previous results and what can be used to mitigate the negative impact of correlation. Then, based on these results, we study whether location-hiding is feasible with correlated host vulnerabilities.

In the simulation, we partition hosts in the resource pool into multiple domains (recall that hosts in the same domain share correlated vulnerabilities, and hosts in different domains are uncorrelated). We choose  $\lambda_v$  to be close to 1 to represent high correlation<sup>9</sup> – once a host is compromised, other hosts with the same bug (in the same domain) can very likely (with probability  $\lambda_v$ ) be compromised within the next simulation time step.  $\lambda_0$  is set according to Table II-2; other parameters are relative to  $\lambda_0$ , and can be easily inferred. For example, if attackers take ten days to compromise a host, and the proxy migration rate  $\mu_r=10\lambda_0$ , then a proxy migrates once per day (i.e. ten times more frequent than host compromises).

---

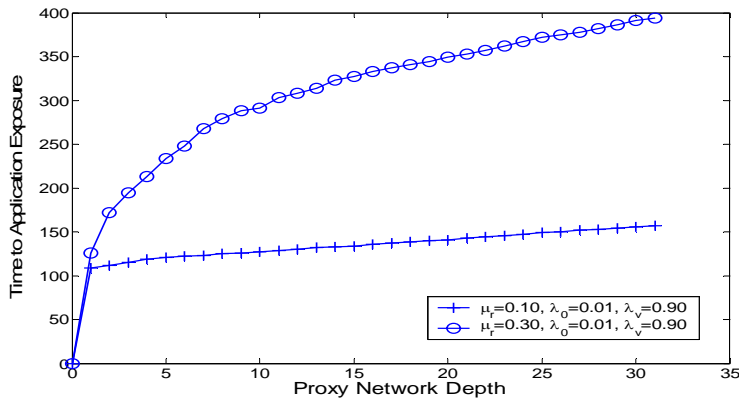
<sup>9</sup> As long as  $\lambda_v$  is significantly larger than  $\lambda_0$  the results are qualitatively the same.

### A) How Does Correlation Affect Previous Results?

To answer this question, we consider a high correlation in host vulnerabilities ( $\lambda_v=0.9$ ). Hosts do not proactively install security patches ( $\mu_s=0$ ), but reactive recovery is “perfect” (all compromised hosts are immediately recovered,  $\rho=1$ ,  $\mu_d=1$ ). Figure 8 shows time to application exposure (unit is simulation time step) as a function of proxy network depth for cases with high proxy migration rates ( $\mu_r=10\lambda_0$  and  $\mu_r=30\lambda_0$  respectively).

Recall that if host vulnerabilities are uncorrelated, time to application exposure grows exponentially with proxy network depth as in Figure 5. However, in Figure 8 the curve stays flat, and location-hiding is infeasible. Therefore, correlated vulnerabilities qualitatively change location-hiding.

The reason is straightforward: with high correlation in host vulnerabilities, the system is nearly continuously in a high compromise regime. Without proactive reset (security patch of known vulnerabilities), the probability of host compromises is almost always high ( $\lambda \approx 0.9$ ). The seemingly high proxy migration rates ( $\mu_r/\lambda_0$  is 10 and 30 respectively) become insignificant in comparison to the correlated host vulnerabilities, thereby providing little defense.

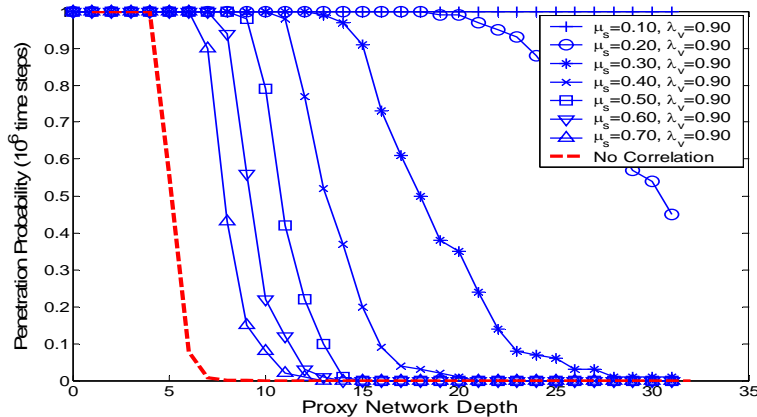


**Figure 8 Time to Application Exposure (with correlated host vulnerability)**

## B) What Mitigates the Impact of Correlation?

Unless the negative impact of correlation can be mitigated, proxy networks cannot achieve location-hiding. We consider two factors for mitigation: proactive reset and host diversity. Proactive reset can patch known vulnerabilities before attacks happen, thereby mitigating the impact of correlation. Meanwhile, host diversity can limit the impact of correlation, since correlated host vulnerabilities only affect hosts inside the same domain.

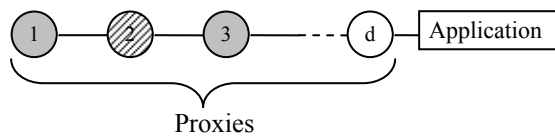
Figure 9 shows how much proactive resets can mitigate the negative impact of correlated host vulnerabilities. It shows the probability to penetrate a proxy network of certain depths within  $10^6$  time steps. Each curve corresponds to a proactive reset rate ( $\mu_s$ ). The uncorrelated case is also plotted for comparison, showing the contrast to the uncorrelated case. Even for high reset rates, impact of correlation is still prominent. This is because proactive resets are not guaranteed to happen before attacks. Therefore proactive resets by themselves cannot effectively contain the impact of correlation.



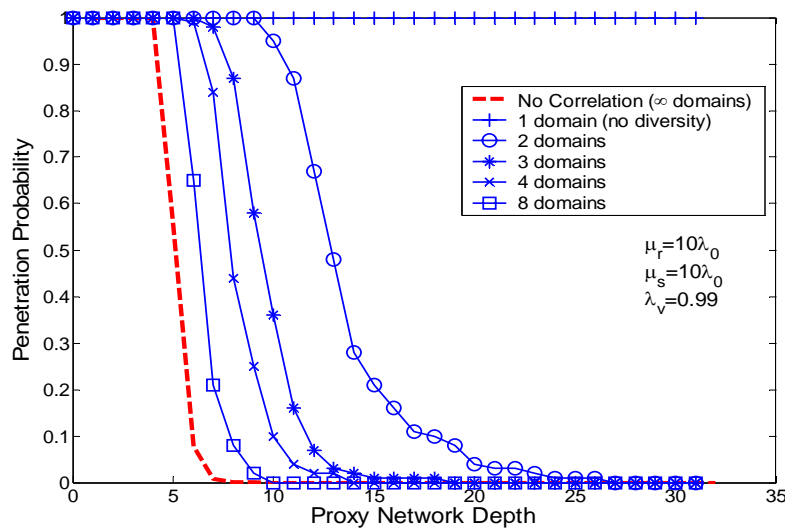
**Figure 9 Impact of Correlated Vulnerabilities (Varying Proactive Reset Rate)**

Host diversity can be used to mitigate the impact of correlation. Consider the case where hosts are in a number of domains; within a domain, hosts have correlated vulnerabilities; across

domains hosts are uncorrelated. Consider a proxy path shown in Figure 10; proxy 1 and 3 run on hosts in the same domain, while proxy 2 runs on a host in a different domain. After proxy 1 being compromised, all the hosts in its domain (including the one hosting proxy 3) become vulnerable. But proxy 2 is not affected and becomes a barrier to slow down attackers. By the time attackers reach proxy 3, there is a good chance that proactive resets have already patched that host, and made it no longer vulnerable. Therefore combining host diversity with proactive resets may effectively contain the impact of correlation.



**Figure 10 Diversity in Proxy Path**



**Figure 11 Impact of Correlated Vulnerabilities (Varying Host Diversity)**

Figure 11 shows that host diversity can mitigate the impact of correlation. In this case, hosts are divided evenly into multiple domains, and proxies migrate randomly across all hosts. Each curve corresponds to a specific degree of host diversity (number of domains). The uncorrelated case is also plotted for comparison. The difference between each curve and the uncorrelated case

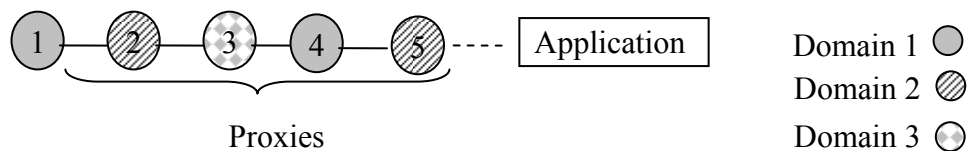
indicates the impact of correlated host vulnerabilities. It turns out that even small degrees of host diversity can be an effective barrier to resist attacks, enabling proactive resets to be applied in time to the vulnerable hosts, and thereby mitigate the impact of correlated vulnerabilities.

### C) Is Location-Hiding Feasible?

We have shown that diversity and proactive resets can potentially mitigate the negative impact of correlated vulnerabilities. However, a naïve scheme (as shown in Figure 11) is insufficient to remove the negative impact of correlation. It is desirable to have a proxy network which can effectively hide location despite the correlated host vulnerabilities. Here we explore how to construct such a proxy network. There are a few observations.

First, placing proxies in randomly chosen domains is suboptimal. If neighboring proxies are in the same domain, their vulnerabilities are correlated, and they will fail together. A better approach is to put neighboring proxies in different domains, increasing their effectiveness on slowing down attacks.

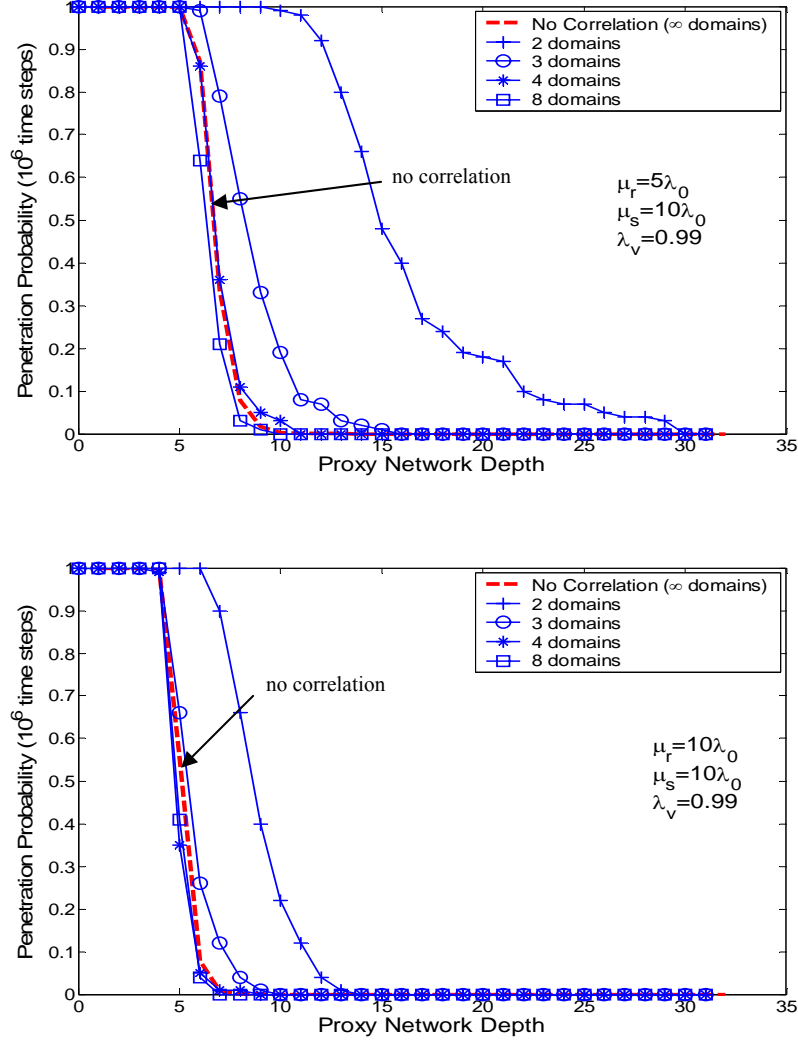
Second, allowing proxies to migrate to random hosts may help attackers, since a proxy may migrate to a vulnerable host which has not been patched yet, and thus undo the designed neighboring property described above.



**Figure 12 Interleaved Design**

We consider an interleaved proxy network design (see Figure 12), where 1) proxies are placed into different domains, maximizing the distance (overlay hop count) between any pair of proxies in the same domain, and 2) proxy migrations are confined to the same domain. For

example, consider a proxy path as shown in Figure 12 and a resource pool with  $k$  domains, proxies on the path can be placed into the  $k$  domains in a round-robin order<sup>10</sup>.



**Figure 13 Impact of Correlated Vulnerabilities (Interleaved Design)**

Figure 13 shows the probability to penetrate a proxy network with various depths within  $10^6$  time steps with different proxy migration rates ( $\mu_r = 5\lambda_0$  and  $\mu_r = 10\lambda_0$ ) for our proxy network design. In both cases, even with a high correlation ( $\lambda_v = 0.99$ ), a small amount of diversity (e.g. 4 domains)

<sup>10</sup> Here we only consider simple proxy network topologies, such as a line or a tree, in which round-robin assignment can trivially solve the problem. Complex topologies may require more sophisticated assignment schemes, which are beyond the scope of this paper.

can almost completely contain the impact of correlated vulnerabilities. To further support this claim, we studied the system for  $10^5$ ,  $10^6$ ,  $10^7$  and  $10^8$  time steps respectively (see Figure 16 in Appendix III). We found that, within the observed ranges ( $10^5 \sim 10^8$  time steps), with 4 or more domains, a system behaves almost identically to one with uncorrelated vulnerabilities (the ideal case). This indicates that using a small degree of host diversity, e.g. 4 domains, the interleaved design can reduce the negative impact of correlated host vulnerabilities significantly, and enable a proxy network to resist penetration attacks effectively.

Here is why a small degree of host diversity can provide effective defense. In the interleaved design for a chain of proxies in a system of  $k$  domains (illustrated in Figure 12), between any two proxies (denoted by A and B) in the same domain, there is a path of  $k-1$  proxies in 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  $k-1$ ) for an uncorrelated proxy path, a small degree of host diversity (or the number of domains  $k$ ) can provide a large penetration time<sup>11</sup>, 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.

### C. *Summary*

From the study of location-hiding problem, we have the following insights.

First, without reconfiguration, a proxy network cannot securely hide application location, and is vulnerable to penetration attacks. Indirection and randomness in proxy networks cannot guarantee location-hiding. So existing approaches [1-4] are vulnerable to penetration attacks.

---

<sup>11</sup> It takes 100 times longer to penetrate a path with length three (case of 4 domains) than length one (case of 2 domains), when  $\mu_r=10\lambda_0$ .

Second, by adding reconfiguration mechanisms, such as proxy migration, location-hiding becomes feasible. Proxy network depth and proxy migration rates are critical factors in location-hiding.

Third, correlated host vulnerabilities can void location-hiding; however, we can exploit host diversity and intelligent proxy network construction to effectively contain the negative impact of correlation, and achieve location-hiding.

Two logical steps support these results. First, we mathematically analyze the situations with uncorrelated host vulnerabilities. Then, we study the general case where host vulnerabilities are correlated. In this case, host diversity is needed to resist attacks effectively. We show how host diversity can be exploited to construct a proxy network which can contain the negative impact of correlation, and behave as well as the uncorrelated case. So in both cases, we show that it is feasible to construct a proxy network to achieve location-hiding, and characterize the circumstances under which it is possible.

#### IV. DISCUSSION AND RELATED WORK

We first discuss several additional important issues to the location-hiding problem, and then summarize the most closely related work.

**Proxy Migration** is a key defensive mechanism discussed in the paper. To explore its feasibility, we implemented a prototype proxy network supporting migration. Our proxy network supports generic TCP applications (we run Apache and http clients in our experiments). Application traffic is tunneled through our proxy network. Proxies are implemented as simple packet switches, and therefore they do not contain hard state. As a result, it is fairly straightforward to implement migration. There are at least two ways to achieve migration. In the first approach, a trusted party creates a new proxy instance at the new location (migration destination), and then disable the old

instance at the original location. The advantage of this approach is that the old proxy is not actively involved in the migration process; therefore even if the old proxy is compromised, the attacker still cannot prevent it from migrating to a new location. However, the in-flight packets on the old proxy might be lost, and thus may cause temporary service disruptions. The second approach is similar to the first approach, except that the old proxy instance transfers all the in-flight packets to the new instance before being disabled. This requires cooperation from the old instance, but the main advantage is that there is no service disruption during migration. We have implemented both mechanisms, and measured the migration cost on a local area network. The overhead is on the order of milliseconds. Such an overhead is negligible for daily or hourly migrations, i.e. “high rate” migration ( $10x \sim 100x$  higher than host compromise rate  $\lambda_0$ ) discussed in our analysis. This indicates that current technology can sufficiently support the proxy migration mechanism we need. We are using this implementation for further experiments. However, how to build an attack-resilient migration scheme to prevent attackers from disrupting the migration process is a separate problem. This involves a different attack which is specific to the migration protocol and is beyond the scope of this paper.

**Performance Implications** of a proxy network. We showed that proxy network depth is a key to location-hiding. However, large proxy network depth may incur performance penalty to the application. Appropriate network depths depend on the security and performance requirement of the application. Our study here provides a framework to reason about design choices of proxy network depth based on the security requirement. Meanwhile, in a separate study, we are investigating the performance implication of proxy networks [15]. Our results show that the performance penalty is not a simple linear function of proxy network depth. If proxies are not completely randomly deployed, but rather following certain simple performance heuristics, proxy

networks do not necessarily incur significant performance penalty, and in many cases may even improve the performance of TCP-based applications. Details of this study can be found in [15].

**Overhead of Defensive Mechanisms** is also an important issue for the use of proxy networks in practice. Specifically, in order to understand the feasibility of location-hiding, we need to evaluate the cost associated with the defensive mechanisms required for location-hiding, such as proxy migration and proactive reset. This cost can be characterized by the frequency of such activities. In the paper, we have shown that, under realistic assumptions about attack speed (e.g. using numbers in Table II-2), a daily proxy migration rate and a daily proactive reset rate are sufficient to keep a proxy network impenetrable. From our proxy network implementation, we demonstrated that the proxy migration overhead is negligible for a daily migration rate. Furthermore, current system management technologies, such as [16, 17], automate the proactive reset process, thereby making it feasible to support daily proactive reset without incurring significantly overhead. These facts indicate that the required defensive mechanisms for location-hiding are realistic in practice.

**Proxy Network Topology** has an important impact on location-hiding. Here, we focus on the feasibility of location-hiding and use a simplified proxy network topology in our analysis, showing that depth is a key factor. However, other properties of the topology, such as connectivity (e.g. vertex degree and neighborhood expansion property), also have qualitative impact. In [18], we use a similar framework to study the impact of overlay network topology on location-hiding from a graph theoretic perspective. We find that excessive connectivity in overlay topology is detrimental to location-hiding; and popular overlays such as Chord [19] are not favorable topologies for location-hiding due to their rich connectivity. We also presented a set of graph theoretic methods to determine whether a topology is favorable for location-hiding.

Many researchers are exploring the use of overlay networks to resist DoS attacks. Secure Overlay Services (SOS) [4] protects applications against flooding DoS attacks by installing filters around applications and only allowing traffic from secret “servlets”. SOS uses Chord [19] to implement communication between users and the secret servlets without revealing the IP addresses of the servlets. Mayday [3] generalizes the SoS architecture and analyzes the implications of choosing different filtering techniques and overlay routing mechanisms. Both SOS and Mayday fit well in the generic proxy network framework studied in this paper. Because of the filtering, servlets play the role of applications, and are the subject of location-hiding. Secrecy of servlets’ location is the key to such approaches, since once servlets’ addresses are revealed, attackers can either bypass the filters to attack the application directly<sup>12</sup> or attack the servlets directly. Internet Indirection Infrastructure (i3) [1, 2] also uses the Chord overlay network to hide an application’s location. It fits well into our generic proxy network framework. However, the work on i3 [1, 2] does not study penetration attacks and or the feasibility of location-hiding – they focus on other issues. None of these approaches (SOS, Mayday and i3) employ active reconfiguration mechanisms which can invalidate the information attackers have.

The main distinctions between our work and these other efforts (SOS, Mayday, and i3) are the following. First, each of these efforts focuses on their specific (Chord-based) implementations. The evaluation of one applies only to that particular implementation. There has been no systematic exploration of the fundamental capabilities and limitations of the general class of proxy network-based location-hiding. Second, these efforts have not studied penetration attacks, which are critical threats to the proxy network-based location-hiding. In [4], the authors focused on flooding DoS

---

<sup>12</sup> In SOS, only traffic from the secret servlets is allowed to pass the filters to reach the application. However, if the IP addresses of the servlets are revealed, attackers can spoof the source addresses of the attack packets to be those of the servlets, and bypass the filters.

attacks on proxy nodes, and studied the connectivity between users and the application during such attacks. In [1, 2], the authors analyzed attacks specific to the i3 protocol; their analysis is not general applicable to other proxy network implementations. Our results show that, in their existing form, these approaches are vulnerable to penetration attacks, and cannot achieve location-hiding. By adding reconfiguration, however, they can potentially achieve location-hiding.

Our work differs from efforts to resist worm-style broad attacks [20], whose goal is to compromise all or at least large fractions of a resource pool. In contrast, our framework models attacks which exploit the indirection structure of the proxy network. We do assume that some other mechanisms are used to ensure the entire resource not to be compromised. The rationale for this is if that problem cannot be solved, little else matters.

Our work here focuses on how to hide application location. Interestingly a complementary problem, hiding user identity, has been well studied since the early 1980's. The solutions range from the early mix email server [21], to distributed Onion Routing schemes [22], and to the more recent Peer-to-Peer schemes such as Tarzan [23] and Pasta [24]. A key difference between the two problems is that there are many users in the system while there are only a handful of applications. Most of the schemes are based on the idea of mixing all input from all users so that an outsider cannot associate a particular message to a particular user. Another key difference is that, in the user-hiding problem, user initiates the communication. As a result, some schemes, such as Onion Routing [22], require senders to construct a route to the receiver before hand. This is infeasible in the location-hiding problem. These key differences make the two problems incomparable, and solutions in that area do not apply directly to application location hiding.

## V. SUMMARY AND FUTURE WORK

We develop a generic framework for proxy network approaches to location-hiding, and use it to study several fundamental properties of the proxy network-based location-hiding. We show that

- existing approaches employing static structure [1-4] are infeasible against penetration attacks,
- by adding proactive defenses, such as proxy network reconfiguration and migration, location-hiding can be feasible. Proxy network depths and reconfiguration rates are the critical factors for effectiveness.
- in many cases, correlated host vulnerabilities can make proxy network-based location-hiding infeasible.
- by exploiting the host (OS/software) diversity and intelligent construction of proxy networks, the negative impact of correlated vulnerabilities can be mitigated, and location-hiding can be achieved.

**Future Work** We have implemented a prototype proxy network to do experiments which corroborate our results and to explore other aspects of the proxy network approach. In particular, we will study the performance implication on such proxy networks, and empirically study proxy networks' capability to sustain connectivity between users and applications during DoS attacks.

Because there are many open questions about location-hiding with proxy networks, we have employed simple models. This is both for tractability and to get broad results. Several aspects of the model can be extended, including models for attacks, defenses and correlated host vulnerability.

The attack model considered here includes only host compromise attacks, and thus does not characterize other forms of attacks, such as traffic analysis and non-technical attacks (e.g. social engineering). It does not include mass attacks, which target on the resource pool rather than exploiting proxy network connectivity structure. Furthermore, currently attacks are modeled as

stochastic trials, which do not capture the fine-grained behavior of attacks. More sophisticated models for defenses and correlation in host vulnerability can be developed based on current models to characterize more realistic correlated host vulnerabilities and reconfiguration mechanisms other than proxy migration, such as more intelligent proxy migration. Future work might address these limitations.

### Acknowledgement

This work is supported in part by the National Science Foundation under awards NSF Cooperative Agreement ANI-0225642 (OptIPuter), NSF CCR-0331645 (VGrADS), NSF ACI-0305390, and NSF Research Infrastructure Grant EIA-0303622. Support from the UCSD Center for Networked Systems, BigBangwidth, and Fujitsu is also gratefully acknowledged.

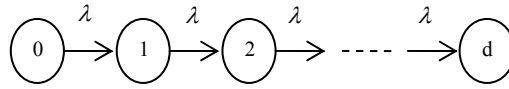
### References

1. Adkins, D., et al., Towards a More Functional and Secure Network Infrastructure. 2003, Computer Science Division, UC Berkeley: Berkeley
2. Adkins, D., et al. Taming IP Packet Flooding Attacks. in *HotNets-II*. 2003.
3. Andersen, D.G. Mayday: Distributed Filtering for Internet Services. in *4th Usenix Symposium on Internet Technologies and Systems*. 2003. Seattle, Washington.
4. Keromytis, A.D., V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. in *ACM Special Interest Group on Data Communications (SIGCOMM)*. 2002. Pittsburgh, PA: ACM.
5. Akamai, Akamai Technology Overview, <http://www.akamai.com/en/html/technology/overview.html>.
6. Hinrikus, T., Skype Application Programming Interface, 2004, <http://www.skype.com/community/devzone/Skype%20API%20description%201.2.pdf>.
7. Cohen, B., Incentives Build Robustness in BitTorrent, 2003, <http://www.bittorrent.com/bittorrentecon.pdf>.
8. Arbaugh, W.A., W.L. Fithen, and J. McHugh, Windows of Vulnerability: A Case Study Analysis". *IEEE Computer*, 2000. **33**: p. 52-59.

9. Browne, H.K., et al., A Trend Analysis of Exploitations. Proceedings of the 2001 IEEE Symposium on Security and Privacy, 2001.
10. Littlewood, B., Predicting software reliability. *Phil. Trans. R. Soc.*, 1989. **327**: p. 513-527.
11. Adams, E.N., Optimizing preventive service of software products. *IBM Journal of Research and Development*, 1984. **28**(1): p. 2-14.
12. Microsoft, Microsoft Security Bulletin, 2004, Microsoft Corporation, <http://www.microsoft.com/technet/>.
13. Lippmann, R., et al., The 1999 DARPA Off-Line Intrusion Detection Evaluation. 2000, MIT Lincoln Lab
14. Lippmann, R.P., et al. Evaluating Intrusion Detection Systems: the 1998 DARPA Off-Line Intrusion Detection Evaluation. in *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*. 2000.
15. Wang, J., X. Liu, and A.A. Chien. Empirical Study of Tolerating Denial-of-Service Attacks with a Proxy Network. in *USENIX Security Symposium'05*. 2005.
16. Sherman, A., et al. ACMS: The Akamai Configuration Management System. in *the 2nd Symposium on Networked Systems Design & Implementation (USENIX NSDI05)*. 2005.
17. HP Open View - Computer and Network Management, <http://www.managementsoftware.hp.com/>.
18. Wang, J., L. Lu, and A.A. Chien. Tolerating Denial-of-Service Attacks Using Overlay Networks – Impact of Topology. in *2003 ACM Workshop on Survivable and Self-Regenerative Systems*. 2003. Washington DC: ACM.
19. 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.
20. Nicol, D.M. and M. Liljenstam, Models of Internet Worm Defense. 2004, Coordinated Science Laboratory; Department of Electrical and Computer Engineering, University of Illinois, Urbana-Champaign
21. Chaum, D.L., Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 1981. **24**(2): p. 84-90.
22. Reed, M.G., P.F. Syverson, and D.M. Goldschlag, Anonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communication Special Issue on Copyright and Privacy Protection*, 1998.
23. Freedman, M.J., et al. Introducing Tarzan, a Peer-to-Peer Anonymizing Network Layer. in *1st International Workshop in Peer-to-Peer Systems (IPTPS'02)*. 2002. Cambridge, Massachusetts.
24. Elnikety, S., et al., Pasta: Anonymous Peer-to-Peer Email System. 2002, Rice University

## APPENDIX I. PROOF OF THEOREM I

If there are no reconfiguration mechanisms which can invalidate the information attackers have, once a proxy becomes exposed, it will remain so. Consider a proxy network of depth  $d$ . Let  $\lambda$  be the probability for a successful host compromise in one stochastic trial. The Markov state transition graph for the system is shown in Figure 14. Node  $i$  ( $0 \leq i \leq d$ ) corresponds to the state where the deepest exposed proxy is at depth  $i$ . Initially, system is at state 0, because edge proxy is exposed.

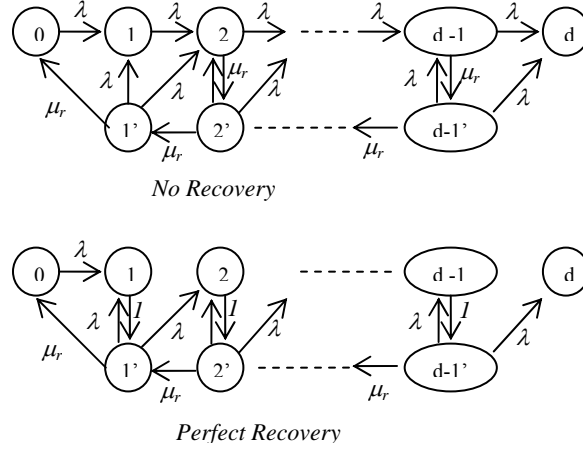


**Figure 14 Markov State Transition (without reconfiguration)**

We first consider the case where this is only one attacker. Let  $p_d(t)$  be the probability of the system reaching state  $d$  before time  $t$ , which is the case where attackers penetrate the proxy network and expose a node at depth  $d$ . It is straightforward to see that  $p_d(t)$  follows an Erlang distribution (each state transition to the right in Figure 14 can be viewed as a Poisson event with rate  $\lambda$ , therefore reaching state  $d$  is equivalent to occurrence of the  $d$ th Poisson event with rate  $\lambda$ ). The expected time to application exposure  $T = d\lambda^{-1} = dT_\lambda$  ( $T_\lambda = \lambda^{-1}$ ).

In the general case, where there are multiple attackers, the expected time to application exposure can only be shorter. Therefore the time to application exposure  $T$  is  $T \leq dT_\lambda$ . ■

**APPENDIX II. PROOF OF THEOREM II**



**Figure 15 Markov State Transition (with migration)**

We consider a chain of proxies with depth  $d$ . Each proxy on the chain is labeled with its depth, e.g. edge proxy is proxy  $0$ , and a proxy at depth  $k$  is proxy  $k$ .

We assume attackers can concurrently attack all the exposed proxies. Markov state transition graph is shown in Figure 15. In state  $0$ , only the edge proxy is exposed. In state  $k$  ( $1 \leq k \leq d$ ), the  $(k-1)$ th proxy is compromised and the  $k$ th proxy is exposed. In state  $k'$ , the  $k$ th proxy is exposed, but the  $(k-1)$ th proxy is not compromised. We study the expected time from state  $0$  to reach state  $n$  in the two boundary scenarios: no recovery and perfect recovery. When there is no recovery, a proxy will stay compromised until it migrates. With perfect recovery, hosts are instantaneously recovered (in Figure 15, state  $k$  goes to state  $k'$  with probability  $1$ ).

**(A) No Recovery**

$T_k$  denotes the expected time to reach state  $d$  from state  $k$  ( $k \leq d$ ). Obviously,  $T_d = 0$  and we want  $T_0$ .

From the state transition graph, we can get

$$\begin{cases} T_0 = 1 + \lambda T_1 + (1 - \lambda) T_0 \\ T_1 = 1 + \lambda T_2 + (1 - \lambda) T_1 \\ T_{1'} = 1 + \lambda(T_2 + T_1) + \mu_r T_0 + (1 - 2\lambda - \mu_r) T_{1'} \\ T_k = 1 + \lambda T_{k+1} + \mu_r T_{k'} + (1 - \lambda - \mu_r) T_k \quad (k > 1) \\ T_{k'} = 1 + \lambda(T_{k+1} + T_k) + \mu_r T_{k-1'} + (1 - 2\lambda - \mu_r) T_{k'} \end{cases} \quad \text{Solve it, we get}$$

$$T_0 = \frac{1}{\lambda} \left( 1 + \frac{\left(\frac{x}{2}\right)^{d-1} - 1}{\left(\frac{x}{2}\right) - 1} + \frac{\left(\frac{x}{2}\right)^{d-1} - 1}{\left(\left(\frac{x}{2}\right) - 1\right)^2} \frac{\left(\frac{x}{2}\right) + 1}{x + 1} - \frac{\left(\frac{x}{2}\right) + 1}{x + 1} \frac{d - 1}{\left(\frac{x}{2}\right) - 1} \right)$$

where  $x = \frac{\mu_r}{\lambda}$ .

### (B) Perfect Recovery

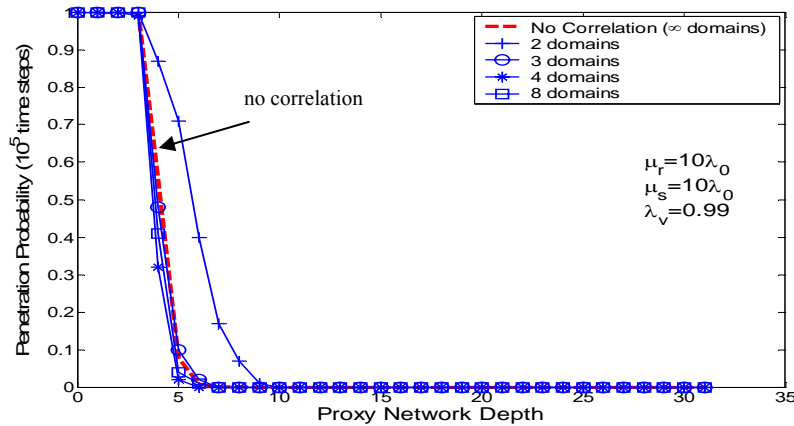
With similar analysis, we get  $T_0 = \frac{1}{\lambda} + \left(1 + \frac{1}{\lambda}\right) \left(\frac{x^d - x}{x - 1}\right) + \left(2 + \frac{1}{\lambda}\right) \left(\frac{x^d - x}{(x - 1)^2} - \frac{d - 1}{x - 1}\right)$ , where  $x = \frac{\mu_r}{\lambda}$ . We can

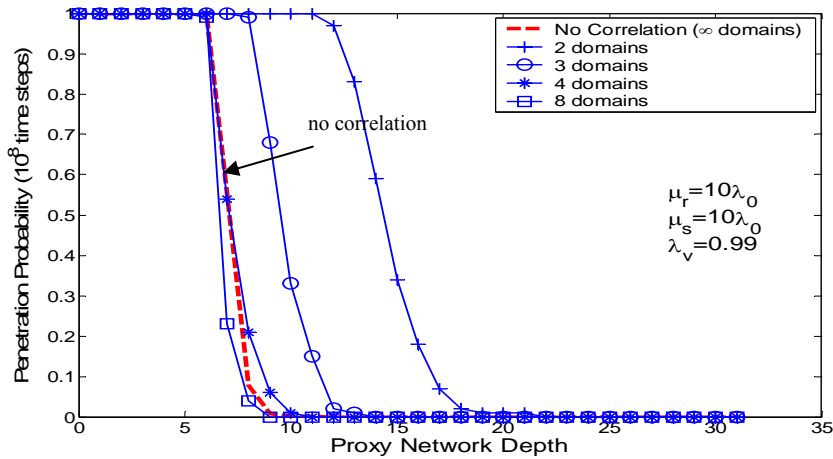
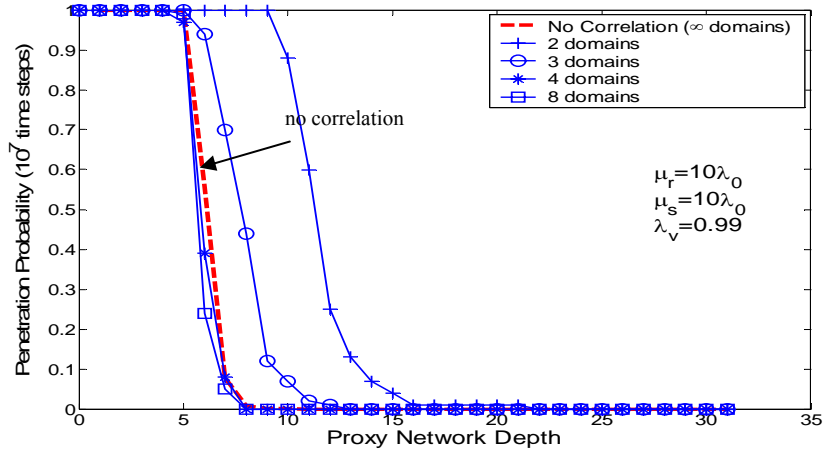
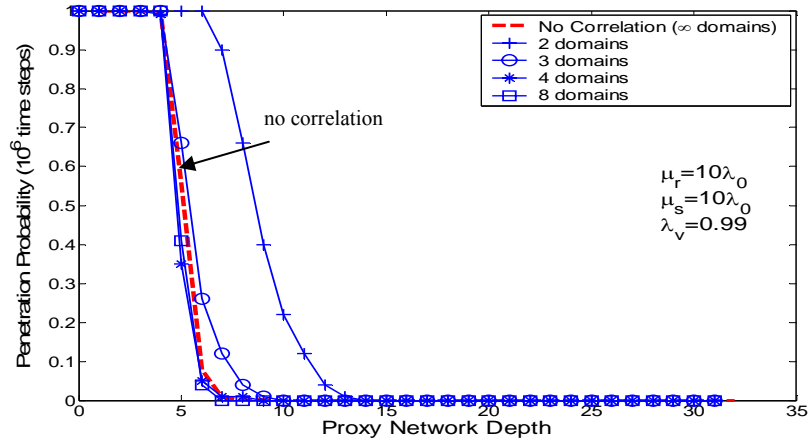
get  $T_0 = \Theta\left(\left(\frac{\mu_r}{\lambda}\right)^{d-1}\right) T_\lambda$  for perfect recovery and  $T_0 = \Theta\left(\left(\frac{\mu_r}{2\lambda}\right)^{d-2}\right) T_\lambda$  for no recovery, where  $T_\lambda = \lambda^{-1}$ . ■

We know that  $T_0$  is between  $\Theta\left(\left(\frac{\mu_r}{2\lambda}\right)^{d-2}\right) T_\lambda$  and  $\Theta\left(\left(\frac{\mu_r}{\lambda}\right)^{d-1}\right) T_\lambda$ . Therefore Theorem II follows. ■

## APPENDIX III. SUPPORTING PLOTS

The following plots show the probability to penetration a proxy network of various depths within  $10^5$ ,  $10^6$ ,  $10^7$  and  $10^8$  time steps respectively. They strongly indicate that by exploiting the diversity, the system behaves very similarly to the uncorrelated case and effectively achieve location-hiding.





**Figure 16 Impact of Correlated Vulnerability (Interleaved Design) (data points observed from  $10^5$  to  $10^8$  time steps)**