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. This framework is used to analyze the effectiveness of location-hiding: characterizing how attacks, defenses, and correlated host vulnerabilities affect feasibility.*

We find that existing approaches employing static structure (e.g. SOS and I3) cannot hide application location because attackers gain information monotonically and in a short period of time penetrate the proxy network. However, we find that adding defenses such as network reconfiguration or migration, which invalidate the information attackers have, makes location-hiding feasible for resisting penetration attacks. We also characterize when proxy networks are effective and stable in location-hiding. In these systems, proxy network depth and reconfiguration rates are critical factors for effectiveness. These results provide both 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 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 indirection. This capability can support anonymous communication, protect applications and hosts from direct attacks, and supporting physical infrastructure 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 application's supporting physical infrastructure[1].

However, some fundamental questions about such proxy networks remain unclear, especially with the presence of intelligent attackers: Can proxy networks achieve location-hiding 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 introduce a stochastic model to characterize the dynamic behavior of the system — particularly how attacks and defense mechanisms affect it. Using this framework and model, we analyze the behavior of proxy network systems and answer some of those questions.

We focus on the classes of attacks that can exploit the indirection structure of the proxy networks and reveal the hidden location, in particular, the class of host compromise attacks and their directed penetration of the proxy network, because such attacks are inherit threat to the essence of the proxy network approach. On the other hand, we do not focus on attacks tied to a specific instance of the proxy network (e.g. SOS), because the goal is to understand the fundamental properties of the generic proxy network rather than a particular instance. Furthermore, our model does not capture the possibility that attackers can attack the entire resource pools (it may be too large, as in the entire internet). For example, we exclude the possibility that all hosts in the resource pool (and therefore all proxies) are under attack, as in that case, location hiding is moot. The key to the validity of this assumption is that it must be difficult for attackers to identify proxy nodes using brute force scan techniques.

Subject to host compromise attacks, we have the following results:

- existing approaches employing static structure [1-4] cannot hide application location against host compromise attacks because attackers gain information monotonically and can in a short period of time penetrate the proxy network.
- by adding defenses such as proxy network reconfiguration and migration which invalidate the information attackers have, location-hiding becomes feasible. In these systems, proxy network depth[2] and

[1] SOS[4] and Mayday[3] hide "secret servlets" instead of applications. See Section IV for detail.

[2] 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.B.

reconfiguration rates are critical factors for the effectiveness of location-hiding.

- correlation in host vulnerabilities can significantly undermine location-hiding; however, by exploiting the limited host (OS/software) diversity, we can effectively contain the negative impact of correlated vulnerabilities and qualitatively improve location-hiding.

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 serves as step stones for future studies based on more complex and realistic models.

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 relates our work to the other studies. We conclude in Section V 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], and then 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 shows a generic proxy network encompassing most of the proposed approaches [1-4]. The essence of the proxy network approach is to allow communication between users and applications without revealing the applications' location. Figure 1 shows a conceptual view of the proxy network from two perspectives; the proxy network is a shield which separates applications from attackers. Applications do not publish their own IP addresses. Instead the addresses of a number of edge proxies are published, and these proxies are used to communicate with the application. In general, the number of edge proxies is a small fraction of the total number of proxy nodes.

B. *System View*

As shown in Figure 1, the system has five key components: applications and users, proxies and hosts, and attackers. Attacks and defenses govern the changes of system states. We first introduce a rigorous terminology and formally describe the system then we discuss changes in system state or dynamics, meanwhile we explain the rationale behind our model.

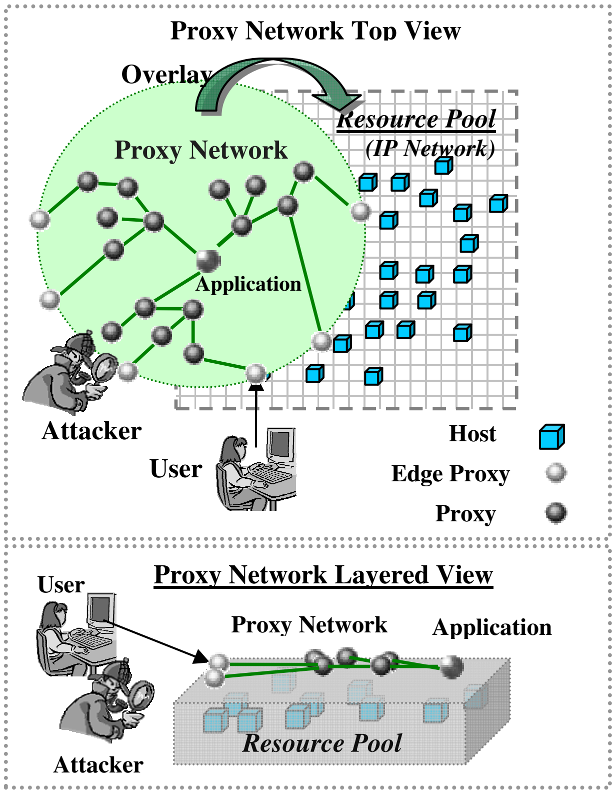


Figure 1 Location-Hiding Proxy Network

1) *System Description and Terminology*

- ***System Components***

Applications are the software implementing application services which respond to user requests (e.g. an e-commerce site). Their location (see definition below) needs to be hidden[3].

Users are the application clients, which are outside the proxy network. They access the applications via edge proxies (see Figure 1) with published locations.

A *proxy* is a software program which forwards application level traffic between applications and users (may through other proxies).

A *host* is an Internet computer with a unique IP address.

If a *node* (either a proxy or an application) runs on a host, that host (or its IP address) is called the *location* of the node. We assume each node has a unique location at any moment (an injective mapping from nodes to hosts). The set of hosts used by proxies constitute a *resource pool*.

A *proxy network* (Figure 1) composed of such proxies and applications form an overlay network. Both proxies and applications are *overlay nodes*. Two nodes are

[3] Some proposals (SOS and Mayday) hide "secret servlets" instead of applications. But our framework still applies. See Section IV for detail.

adjacent or *neighbors* if and only if they know each other's location. We use a graph (*topology graph*) to represent the adjacency structure of the overlay network, where vertices are overlay nodes and edges are drawn between adjacent nodes. The minimal distance (overlay hop count) from edge proxies to the application[4] is the *depth* of the proxy network.

Note that there is a distinction between the adjacency structure of an overlay and its routing structure. The adjacency structure (topology graph) describes how the location information is shared among the overlay nodes, and the amount of information disclosed to attackers once they take control over a proxy as described later; while the routing structure describes how to route a message between edge proxies and the application. They are not necessarily the same in general.

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. However, they are capable of compromising hosts and proxies as described later.

- ***System State***

In the proxy network approach, we assume there is a large resource pool of hosts which are beyond the capability of attackers; so that attackers cannot attack a proxy unless they know its location (massive attacks are discussed in Section V). With this perspective, we define the system state as follows:

The system state consists of the host state and the 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*[5]. 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, because attackers can get the location information from the compromised node. Edge proxies are always exposed, because their location is published for users to access.

2) *Dynamic Changes of System State*

Changes of system state are determined by attacks and defensive mechanisms as described below.

[4]For succinctness, we assume one application per proxy network. Our study can be straightforwardly extended to multiple applications sharing the same proxy network.

[5]Terms "intact" and "compromised" are overloaded for overlay nodes and hosts.

- ***Attacks***

Attackers' goal is to reveal the location information embedded in the proxy network, exploring it adjacency structure, and ultimately expose the application. We consider *host compromise* attacks, an important threat to location-hiding schemes, encompassing a wide range of attacks which can reveal information stored on the victim host or allow attackers to eavesdrop on network traffic by exploiting vulnerabilities of that host or its local network[6]. Other forms of attacks not captured in our framework are considered in Section V. Host compromise attacks can turn intact hosts into compromised hosts. Attackers penetrate proxy networks by compromising nodes along a path from edge proxies towards the application and eventually causing application exposure.

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

- ***Defensive Mechanisms***

There are two defensive mechanisms: resource recovery and proxy network reconfiguration.

<u>*Resource Recovery*</u> 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; furthermore, they also install security patches for all the known vulnerabilities.

<u>*Proxy Network Reconfiguration*</u> changes the location of proxies and the structure of the proxy network. These techniques invalidate the location information acquired by attackers (turning exposed nodes to intact nodes) and disrupt attackers' penetration. We include a simple form of reconfiguration "proxy migration": in which proxies change location but the proxy network topology is unchanged. Proxy migration allows exposed proxies to move to locations unknown to attackers and become intact[7]. In this paper, we study the case where proxies randomly choose hosts from the resource pool during

[6] 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.

[7] 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.

migration, and the overhead of migration is negligible compared to interval between migrations. More complex migration schemes involving intelligent selection of hosts are considered in future work.

C. 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.

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 (operating systems) and have similar configurations, so 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.

1) 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, λ_0 is the rate at which new vulnerabilities in a particular domain are discovered and exploited. λ_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, λ_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, λ_0 and λ_v are the probabilities of turning intact hosts to the compromise state in a single time step.

Furthermore, λ_0 is typically significantly smaller than λ_v ($\lambda_0 << \lambda_v$). Studies on computer vulnerabilities and attack incidents [5, 6] showed that discovery and exploitation of new vulnerabilities is typically hard, requiring a significant amount of time and expertise in 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, because not all compromises are detectable: ρ is the true positive ratio of the detectors (the fraction of all compromises which are detectable), and μ_d is speed of reactive recovery, or the probability of recovering a detectable compromise within a time step. μ_s is the rate of proactive reset. A given host has probability μ_s to be proactively reset within a single time step.

2) Overlay Node State Transition

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

Since we only consider a simple type of migration, where proxies randomly choose hosts to migrate to and the migration overhead is negligible compared to the interval between migrations, we can use one parameter, migration rate, to describe this process. Proxies migrate at rate μ_r, i.e. in each time step, a proxy has probability μ_r to move to a different host. This proxy becomes intact if its new host is intact and none of its neighbors are compromised.

Notation	Meaning
λ_0	Rate of new vulnerability discovery & exploit in a domain
λ_v	Rate of host compromises based on known vulnerability
μ_s	Rate of proactive reset
ρ	True positive ratio of reactive recovery
μ_d	Speed of reactive recovery
μ_r	Rate of proxy migration

Table II-1 Stochastic Model

D. Discussion

Since little is understood about location-hiding, we choose a concise model to keep the problem tractable and to build intuitions. Meanwhile, the model is not overly-simplified; it 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. A key simplification is to use one rate to describe the discoveries of new vulnerabilities (a Poisson rate in the stochastic sense). Previous research [15, 16] on vulnerabilities in large computer systems shows that Poisson process can approximate the discovery of vulnerabilities on large systems, thereby justifying this simplification.

To get a flavor of what this model corresponds to in practice, we collected some numbers from real systems.

λ_0 is the rate of new vulnerability discovery. Microsoft security bulletin [7] catalogues the critical vulnerabilities (remotely exploitable) of Windows Xp Professional and Windows 2K Server (see Table II-2) for the past three years. There are about 20 new vulnerabilities discovered each year, and on average, a new vulnerability is discovered every two to three weeks.

Year	2003	2002-2003	2001-2003
Winxp Pro	19	39	44
Win2k Server	19	43	71

Table II-2 Windows Vulnerability Statistics

λ_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.

ρ is the true positive ratio of intrusion detection systems (IDS), and μ_d is the speed of recovery. Previous research on IDS [17, 18] indicates that modern IDS can achieve ρ over 0.80 and achieve real time detection. Therefore μ_d is primarily determined by how fast a detected intrusion can be removed.

μ_s indicates how frequent a host is cleanly reset. Note that we normally compare it to λ_0 in the analysis, e.g. from Table II-2 we may say that $\mu_s=3\lambda_0$ indicates a host is cleanly reset weekly.

μ_r is the migration rate. Since proxies do not contain persistent data, it is fairly straightforward to implement proxy migration. Our prototype implementation[8] on a local area network has a sub-second migration overhead. This suggests that current technology can well support daily/hourly or higher proxy migration rates, i.e. 10x~100x higher than λ_0.

With the advance of technology (both on the defender side and the attacker side), these numbers may change dramatically. They only help to interpret our analysis in practical settings.

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 study analytically when correlated host vulnerabilities are considered, we approach it in two steps.

First, we study the case of uncorrelated host vulnerabilities ($\lambda_0=\lambda_v$), and provide two theorems to characterize the dynamic system behavior, and show that with appropriate defense, location-hiding is feasible.

[8] 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.

Second, we use a Monte Carlo simulation to study the general case with correlated vulnerabilities. In particular, we will show that it is possible to construct a proxy network which 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 to provide a baseline understanding of the location-hiding problem and to provide a basis for a more general analysis.

We focus on two fundamental questions in location-hiding: Can proxy networks hide location at all? How do the properties of defenses affect location-hiding?

Before addressing these questions, we first present two theorems to quantify the expected time for attackers to penetrate a proxy network and expose the application when host vulnerability is uncorrelated.

Theorem I

Without proxy network reconfiguration, the expected time to application exposure $T\le dT_\lambda$ where T_λ 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 μ_r, which is sufficiently high ($\mu_r\ge 2\lambda$), the expected time to application exposure T grows exponentially with the proxy network depth d; specifically $\Theta((\frac{\mu_r}{2\lambda})^{d-2})T_\lambda \le T \le \Theta((\frac{\mu_r}{\lambda})^{d-1})T_\lambda$, *where λ is the host compromise rate*[9], *and $T_\lambda=\lambda^{-1}$ is the expected time to compromise a host.*

Proofs of Theorem I and II are in Appendix I and Appendix II respectively. Equipped with these theorems, we study the location-hiding problem.

1) *Can Proxy Networks Hide Application Location?*

Theorem I shows that without proxy network reconfiguration, 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 host compromise attacks and cannot achieve location-hiding. Intuitively, without reconfiguration attackers can gain information monotonically. Once a proxy gets exposed, it remains so since then. Therefore attackers only need to compromise all the proxies on a path to the application to cause application exposure.

[9] We use λ instead of λ_0 here to distinguish this uncorrelated case from the general case with correlated vulnerabilities.

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 significantly improve location-hiding. Consequently, proxy networks of moderate depth and reconfiguration can effectively prevent attacker penetration, securely hiding location. For example, if proxy migration rate is ten times greater than host compromise rate, then penetrating a proxy network of depth four takes a thousand times longer than compromising one host, a depth six takes a hundred thousand times longer, eliminating this type of attack as a practical concern.

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

2) *How Do Defenses Affect Location-Hiding?*

There are three key defense properties: proxy network depth, proxy migration rate and resource recovery performance.

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", all compromised hosts are immediately recovered. These plots provide an envelope for general cases with any resource recovery schemes.

Impact of Proxy Network Depth

Figure 2 shows time to application exposure as function of proxy network depth when proxy migration 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). Effectively attacks cannot penetrate proxy networks of modest depth. Thus, proxy networks can be an effective barrier to penetration attacks and achieve location-hiding.

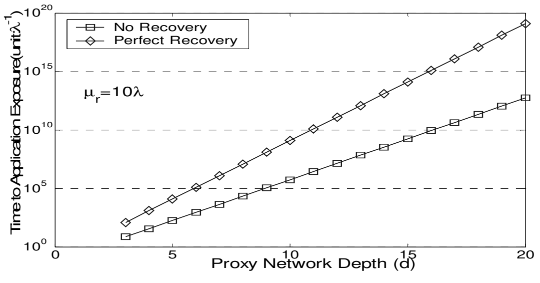


Figure 2 Impact of Proxy Net Depth (fast migration)

Impact of Proxy Migration

Figure 3 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 significantly increases the expected time to application exposure (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 make time to exposure 1000 times longer.

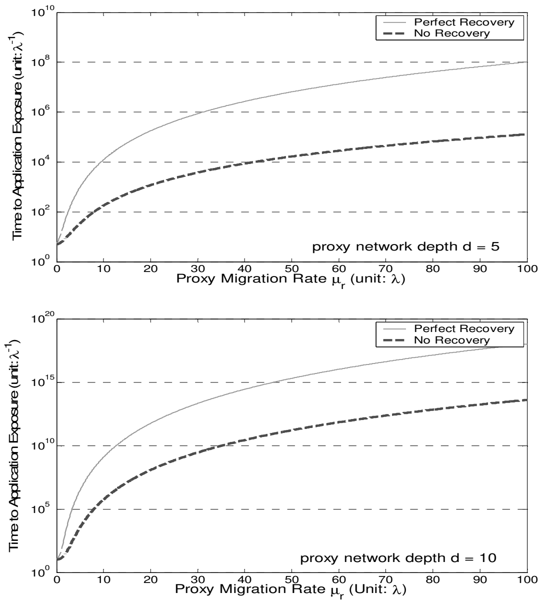


Figure 3 Impact of Proxy Migration

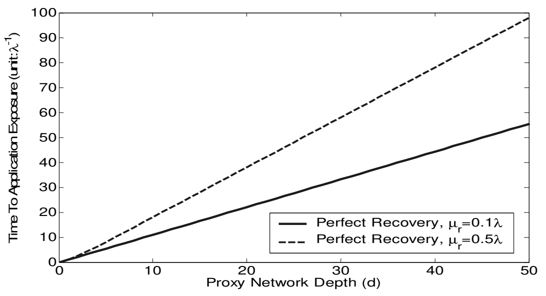


Figure 4 Impact of Proxy Net Depth (slow migration)

More importantly, proxy migration rate has a critical impact that it needs to be sufficiently fast ($\mu_r > 2\lambda$) to guarantee defense against attacks. Figure 4 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 2 and Figure 4 makes it clear that proxy migration is a critical factor in location-hiding, qualitatively changing the effectiveness of location-hiding.

Impact of Resource Recovery

Figure 2 and Figure 3 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 greatly improve location-hiding (both figures show that a perfect recovery scheme can prolong the exposure time several magnitudes over the case without recovery). Furthermore, resource recovery schemes have unique impact to sustain intact host population, and help to overcome the negative impact of correlated host vulnerabilities.

To summarize, both proxy network depth and proxy migrate are critical factors in location-hiding. With a sufficient proxy migration rate, proxy networks can effectively hide locations.

B. Simulation Study: Correlated Vulnerabilities

From the previous section, we know that with proxy migration proxy networks can effectively hide application location -- time to application exposure increases exponentially with proxy network depth. In this section, we use a Monte-Carlo simulation to study a more realistic case where hosts have correlated vulnerabilities.

We first 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 choose λ_v to be close 1 to represent high correlation, i.e. once a host is compromised, other hosts with the same bug can almost surely (with probability λ_v) be compromised within the next simulation time step. Other parameters are relative to λ_0, and can be easily inferred.

1) How Does Correlation Affect Previous Results?

To answer this question, we consider (shown in Figure 5) high correlation in host vulnerabilities (λ_v=0.9). Hosts do not proactively install security patches (μ_s=0), but reactive recovery is "perfect" (all compromised hosts are immediately recovered, ρ=1, μ_d=1). Figure 5 shows time to application exposure (unit is simulation time step, which is approximately one hour) as a function of proxy network depth with high proxy migration rate (μ_r=10λ_0 and μ_r=30λ_0 respectively).

Recall that if the vulnerabilities were uncorrelated, time to application exposure would grow exponentially with proxy network depth as in Figure 2. However, in Figure 5 the curve stays flat and the proxy network cannot hide location at all. Correlated vulnerabilities qualitatively change the location-hiding capability of proxy networks.

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 (μ_r/λ_0 is 10 and 30 respectively) become insignificant in comparison to the correlated vulnerabilities, and provide little defense.

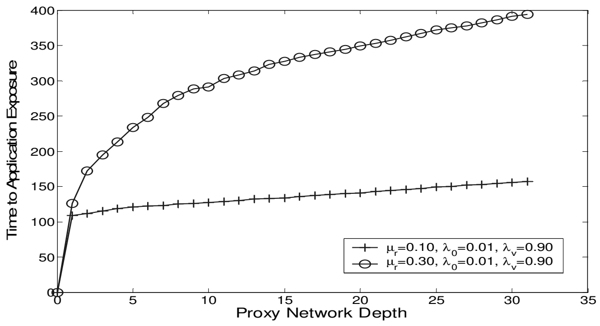


Figure 5 Time to Application Exposure (with correlated host vulnerability)

2) What Mitigates the Impact of Correlation?

Unless the negative impact of correlation can be mitigated, proxy networks cannot achieve location-hiding. We consider two techniques for mitigation: proactive reset and limited 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, because correlated vulnerabilities only affect hosts inside the same domain.

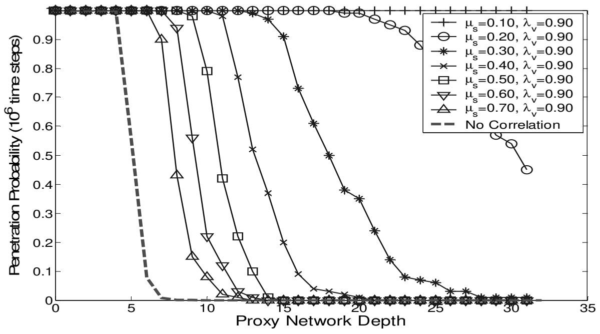


Figure 6 Impact of Correlated Vulnerabilities (Varying Proactive Reset Rate)

Figure 6 shows how much proactive resets can mitigate the negative impact of correlated 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 (μ_s). The uncorrelated case is also plotted for comparison, showing the contrast to the uncorrelated case. Even for extremely high reset rates, impact of correlation is still prominent, with a much larger change of compromise than the uncorrelated case. This is because proactive resets are not guaranteed to happen before attacks, and therefore alone cannot effectively contain the impact of correlation.

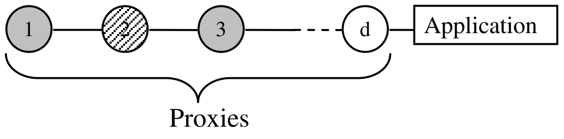


Figure 7 Diversity in Proxy Path

Intuitively, limited host diversity can be used to fight correlation. Consider the case in which hosts are divided into a small set of domains. Within each domain, there are correlated vulnerabilities. Across domains there is no correlation. Consider a proxy path shown in Figure 7; 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 is 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, which is no longer vulnerable. Therefore combining host diversity with proactive resets may effectively containment the impact of correlation.

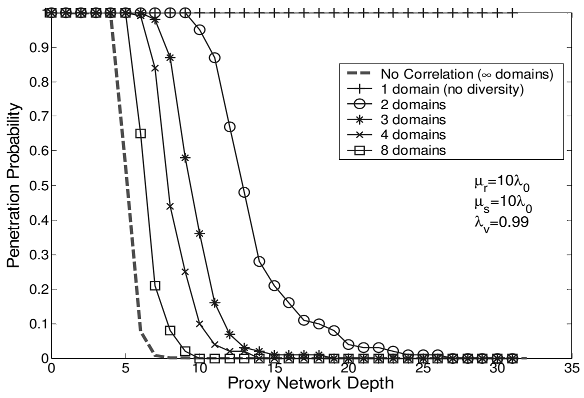


Figure 8 Impact of Correlated Vulnerabilities (Varying Limited Host Diversity)

Figure 8 shows that limited 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 vulnerabilities case is also plotted for comparison. The difference between each curve and the uncorrelated case indicates the impact of correlated vulnerabilities. It turns out 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.

3) 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 8) 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 on hosts in randomly chosen domains is suboptimal. If neighboring proxies run in the same domain, their vulnerabilities are correlated and they will fail together. A better approach is to make sure neighboring proxies are in different domains, increasing their effectiveness on slowing attacks.

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

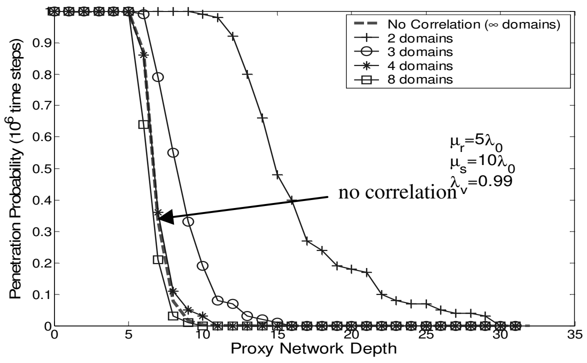
We consider a proxy network design where 1) proxies are placed onto hosts in different domains such that the distance (overlay hop count) between any pair of proxies in the same domain is maximized and 2) proxy migrations are confined to the same domain. For example, consider a proxy path as shown in Figure 7 and a resource pool with k domains; proxies on the path can be placed into the k domains in a round-robin order[10].

Figure 9 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 an high correlation ($\lambda_v=0.99$), a small amount of diversity (4 domains) 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 12 in Appendix III). We found that within the observed ranges (10^5 ~ 10^8 time steps), with 4 or more domains, a system behaves almost identically to one with uncorrelated vulnerabilities (the ideal case). This is remarkable! It indicates that correlated vulnerabilities have been effectively contained.

The intuition is as follows: correlation helps attackers when proxies running in a domain with compromised hosts are exposed. This is because the correlation of vulnerabilities in that domain means that attackers can quickly compromise the proxy and penetrate further. In

[10] 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.

the interleaved design, no two successive proxies run on hosts with correlated vulnerabilities. So the proxies on hosts with uncorrelated vulnerabilities, delay attackers and thereby allow more chances for correlated proxies to install security patches and remove the vulnerability. We know that the time to penetrate an interleaved (uncorrelated) proxy path grows exponentially with path length. So, two domains or even 4 domains can effectively resist an attack[11]. A higher proxy migration rate further improves behavior under correlated vulnerabilities because it slows attackers (see Figure 9). Increasing the proxy migration rate allows correlation to be more effectively contained (curves of the correlated cases are closer to the uncorrelated case).



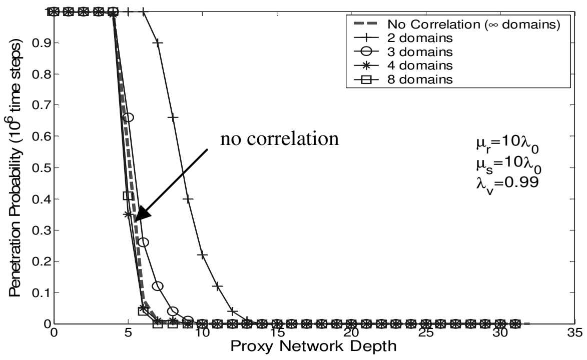


Figure 9 Impact of Correlated Vulnerabilities (Exploiting Diversity)

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. Such a proxy network is vulnerable to host compromise attacks, and can be easily penetrated. Indirection and randomness in proxy networks cannot guarantee location-hiding. This result implies that existing approaches [1-4] are vulnerable to host compromise attacks.

Second, with proxy migration, a simple reconfiguration scheme, proxy networks can effectively hide application location. Proxy network depth and proxy migration rates are critical factors in location-hiding.

Third, correlated host vulnerabilities can significantly undermine location-hiding, however, we can exploit limited host diversity to effectively contain the negative impact of correlated vulnerabilities.

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

IV. RELATED WORK

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 [8] to implement communication between users and the secret servlets without revealing the IP addresses of the servlets, and focuses on how well SOS can keep applications accessible to users when the SOS network is under a flooding DoS attack. 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, because once servlets' addresses are revealed, attackers can either bypass the filters to attack the application directly[12] or attack the servlets directly.

Internet Indirection Infrastructure (i3) [1, 2] also uses the Chord overlay network to hide an application's location, and fits well into our generic proxy network framework. However, the work on i3 [1, 2] does not study host compromise attacks and or the feasibility of location-hiding – they are focused on other issues.

[11] It takes attackers 100 times longer to penetrate a path with length 3 (case of 4 domains) than to penetrate one proxy (case of 2 domains), when $\mu_r=10\lambda_0$.

[12] 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.

None of these approaches (SOS, Mayday and i3) employ active reconfiguration mechanisms which can invalidate the information attackers have. Our results show that in their existing form, these approaches are vulnerable to host compromise attacks and cannot achieve location-hiding. However, by adding reconfiguration, these approaches can potentially achieve location-hiding.

Wang et al. [9] developed a similar framework to study the impact of overlay network topology on location-hiding from a graph theoretic view. They found that excessive connectivity in overlay topology is detrimental to location-hiding; and popular overlays such as Chord [8] are not favorable topologies for location-hiding due to their rich connectivity. They also presented a set of graph theoretic methods to determine whether a topology is favorable for location-hiding. We focus on the feasibility of location-hiding, while [9] studies the impact of overlay topology. The two studies complement each other.

Our work differs from efforts to resist worm-style broad-based attacks [10] whose goal is to compromise all or at least large fractions of a resource pool. In contrast, our framework models attacks to exploit the indirection structure of the proxy network. We do assume that some other mechanisms ensure the entire resource not to be compromised. The rationale for this is if that problem cannot be solved, little else matters. One more critical issue is that the proxies in our approach must be a small fraction of the resource pool, and not be easily identifiable. Otherwise a host scan could quickly narrow the resource pool to those currently hosting proxies.

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 [11], to distributed Onion Routing schemes [12], and to the more recent Peer-to-Peer schemes such as Tarzan [13] and Pasta [14]. 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 user initiates the communication. In some schemes, such as Onion Routing [12], senders need to construct a route to the receiver before hand. 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 networks and use it to study fundamental questions in location-hiding. We find that existing approaches employing static structure (e.g. SOS and I3) cannot hide application location because attackers gain information monotonically and can eventually penetrate the proxy network. However, by adding defenses such as proxy network reconfiguration or proxy migration which invalidates the information attackers have, location-hiding becomes feasible. In these systems, proxy network depth and reconfiguration rates are critical factors for the effectiveness of location-hiding. Furthermore, correlated host vulnerabilities may significantly undermine location-hiding, but limited host diversity can help to effectively contain the negative impact of correlated vulnerabilities.

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). 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 reconfiguration mechanisms other than proxy migration and more realistic correlated host vulnerabilities. Future work might address these limitations.

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 SIGCOMM'02*. 2002. Pittsburgh, PA: ACM.
5. Arbaugh, W.A., W.L. Fithen, and J. McHugh, *Windows of Vulnerability: A Case Study Analysis"*. IEEE Computer, 2000. **33**: p. 52-59.
6. Browne, H.K., et al., *A Trend Analysis of Exploitations*. Proceedings of the 2001 IEEE Symposium on Security and Privacy, 2001.
7. Microsoft, *Microsoft Security Bulletin*. 2004, Microsoft Corporation.
8. Stoica, I., et al. *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications*. in *ACM SIGCOMM'01*. 2001.
9. 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.
10. 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.
11. Chaum, D.L., *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms.* Communications of the ACM, 1981. **24**(2): p. 84-90.
12. 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.
13. 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.
14. Elnikety, S., et al., *Pasta: Anonymous Peer-to-Peer Email System*. 2002, Rice University.
15. Littlewood, B., *Predicting software reliability.* Phil. Trans. R. Soc., 1989. **327**: p. 513-527.
16. Adams, E.N., *Optimizing preventive service of software products.* IBM Journal of Research and Development, 1984. **28**(1): p. 2-14.
17. 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.
18. Lippmann, R., et al., *The 1999 DARPA Off-Line Intrusion Detection Evaluation*. 2000, MIT Lincoln Lab

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 λ be the probability for a successful host compromise in one stochastic trial. The Markov state transition graph for the system is shown in Figure 10. Node i ($0 \le i \le d$) corresponds to the state where the deepest exposed proxy is at depth i. Initially, system is at state 0, because edge proxy is exposed.

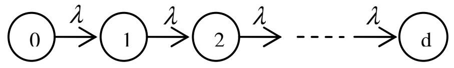


Figure 10 Markov State Transition (without reconfiguration)

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 10 can be viewed as a Poisson event with rate λ, therefore reaching state d is equivalent to occurrence of the dth Poisson event with rate λ). 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 $T \le dT_\lambda$. ▌

APPENDIX II. PROOF OF THEOREM II

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.

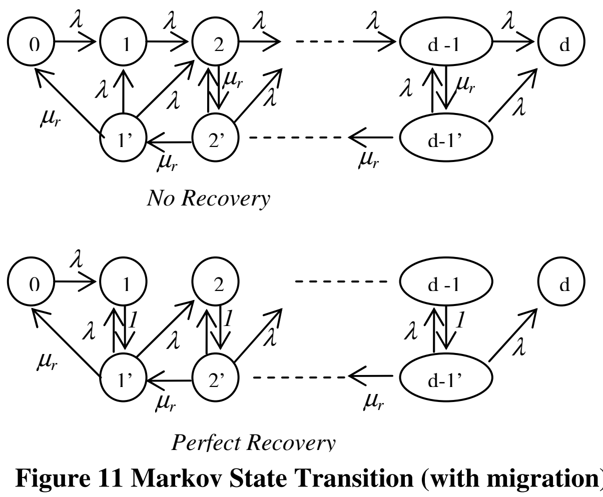


Figure 11 Markov State Transition (with migration)

We assume attackers can concurrently attack all the exposed proxies. Markov state transition graph is shown in Figure 11. In state 0, only the edge proxy is exposed. In state k (1≤k≤d), the (k-1)th proxy is compromised and the kth proxy is exposed. In state k', the kth 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 the state transition graph, 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≤d). Obviously, $T_d = 0$ and we want T_0. From the state transition graph, we can get

$$\begin{cases} T_0 = 1+\lambda T_1 + (1-\lambda)T_0 \\ T_1 = 1+\lambda T_2 + (1-\lambda)T_1 \\ T_{1'} = 1+\lambda(T_2+T_1)+\mu_r T_0 + (1-2\lambda-\mu_r)T_{1'} \\ T_k = 1+\lambda T_{k+1}+\mu_r T_{k'} + (1-\lambda-\mu_r)T_k \qquad (k>1) \\ T_{k'} = 1+\lambda(T_{k+1}+T_k)+\mu_r T_{k-1'} + (1-2\lambda-\mu_r)T_{k'} \end{cases}$$

Solve it, we get

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

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

(B) Perfect Recovery

With similar analysis, we get $T_0 = \frac{1}{\lambda}+(1+\frac{1}{\lambda})(\frac{x^d-x}{x-1})+(2+\frac{1}{\lambda})(\frac{x^d-x}{(x-1)^2}-\frac{d-1}{x-1})$,

where $x = \frac{\mu_r}{\lambda}$. We can get $T_0 = \Theta((\frac{\mu_r}{\lambda})^{d-1})T_\lambda$ for perfect recovery and $T_0 = \Theta((\frac{\mu_r}{2\lambda})^{d-2})T_\lambda$ for no recovery, where $T_\lambda=\lambda^{-1}$. ▌

We know that T_0 is between $\Theta((\frac{\mu_r}{2\lambda})^{d-2})T_\lambda$ and $\Theta((\frac{\mu_r}{\lambda})^{d-1})T_\lambda$. 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.

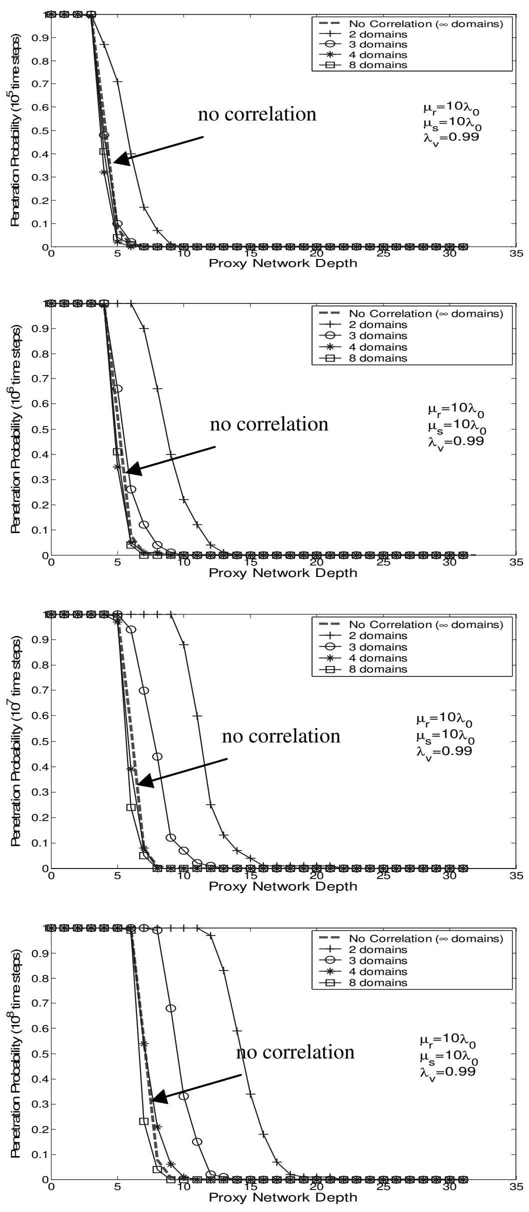


Figure 12 Impact of Correlated Vulnerability (Exploiting Diversity) (data points observed from 10^5 to 10^8 time steps)