Proceedings of the IEEE International Information Assurance Workshop, 2005

Malware Defense Using Network Security Authentication*

Joseph V. Antrosio and Errin W. Fulp
Wake Forest University
Department of Computer Science
Winston-Salem, NC, USA
nsg.cs.wfu.edu
{antrjv1|fulp}@wfu.edu

Abstract

Malware defenses have primarily relied upon intrusion fingerprints to detect suspicious network behavior. While effective for discovering computers that are already compromised, these systems are not designed to stop the spread or damage of malware. Standard gateway firewalls can prevent outside-based attacks; however, they are ineffective in a mobile network where threats originate from inside and administrators have limited control over client machines.

This paper introduces a new strategy for malware defense using security authentication which focuses on vulnerabilities rather than exploits. The proposed system uses a remote security scanner to check for vulnerabilities and quarantines machines using logical network segmentation. This maximizes the usefulness of the machine in question while preventing attacks. Furthermore given the unique ability to quarantine machines ***without any specialized host software****, the proposed system* ***can defend against internal malware threats in a mobile network****. Positive results have been achieved utilizing a proof-of-concept model and standard networking tools.*

[1]This work was supported by the U.S. Department of Energy MICS (grant DE-FG02-03ER25581). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the DOE or the U.S. Government.

1 Introduction

The benefits of highly interconnected networks and systems in recent years have, unfortunately, been accompanied by an increased number of security threats. For example, the damage from malware has been recently estimated at $12.5 billion worldwide in 2003 alone and is expected to increase [5]. Malware is any unwanted software that exploits flaws in other software to gain illicit access. A computer worm is one of the most common forms of malware, and is typically defined as a computer program that replicates independently by sending itself to other systems [2, 3, 7]. This definition is important since a worm, unlike other forms of malware, does not require human interaction such as checking e-mail or transmitting files. In these scenarios, the user is initiating the action, and the machine cannot be compromised independently of this interaction. Therefore, computer worms are among the most dangerous forms of malware and are difficult to defend against.

Despite the large quantity and variety of known worms, only one worm, the Morris Worm exploited a *zero-day* vulnerability. This is a vulnerability that was unknown to the general public, but fortunately these occurrences have been rare [3]. All other worms have been created sometime after the vulnerabilities have been discovered, publicized, and often fixed. Although the threat of a zero-day worm exists, the greater threat continues to be from published vulnerabilities, thus it is important to focus efforts on curtailing the spread of worms that exploit them.

Current malware defenses are largely based on

fingerprint or signature technology which look for a type of network behavior or even specific code [5, 6, 13, 18]. A malware signature or fingerprint is the sequence of network transmissions required to exploit a vulnerability. Signature-based solutions are limited in their effectiveness, as new variants of worms can bypass the malware defense by changing their signature or fingerprint. Although these systems are effective for detecting the spread of known malware, they rely on continuous filtering at higher OSI layers, and thus are very resource intensive. These defense systems are not suited for protecting high speed connections without significantly reducing bandwidth. The resource intensive nature of these defenses prevents them from being implemented at every level in a network and are most often implemented at the slowest connection of the network—the connection to the Internet.

Despite the fact that vulnerabilities are often more publicized than the exploits, past and current research focuses on the attack stage of malware [6, 13, 18]. For instance, researchers at University of Massachusetts at Amherst proposed an early monitoring system for Internet worms [18]. The goal is to detect unknown worms so that the worm's particular behavior can be analyzed and combated by current signature-based defenses. This strategy offers a method for detecting zero-day worms that exploit previously unknown vulnerabilities. However, this strategy is weakened by several factors. Firstly, while current fingerprint-based defense systems would find early detection information useful, it is still difficult to engage these defenses in a small time frame such as that of a quick propagating virus such as SQL Slammer [3] because the signature must be discovered and distributed to all defense systems immediately. Secondly, this monitoring system unfortunately relies on the hundreds and even thousands of machines that are already infected by the worm to generate enough significant traffic to be noticed by the early warning system. This strategy provides no preemptive defense for the thousands of machines already infected by the worm. Thirdly, widescale data mining is crucial for the system's success but there is no current system in place nor feasible plans to collect worldwide traffic data in a single location.

Although a publicized vulnerability often has a fix (software patch) available, inconveniences of human interaction with these fixes can lead to unpatched systems. Since applying patches is the optimal solution for worm defense, there has been research on auto-patching systems and even auto-patch generation for certain kinds of attacks [13]. The possibility of this worm vaccination framework is promising for several reasons. Firstly, it does not rely on global network traffic monitoring. Secondly, the system is easily applied at any location and is not specific to specific software packages. It would also be effective against unknown worms. Despite all of its benefits, however, there is a key assumption inherent in the work. The paper implies that every system in the network is under control of the system administrator which is hardly the case with publicly accessible mobile networks.

Given the difficulty of successfully defending computer systems from attacks, this paper proposes a novel system utilizing *security authentication* for malware defense. The proposed architecture is based on periodically scanning for vulnerabilities and quarantining vulnerable systems using logical network segmentation. Systems are given limited access to the network based on their perceived threat. Commercial systems, such as Perfigo[1] [11], have a similar ability to isolate/quarantine vulnerable devices and provide controlled access to patch servers and remediation systems.

Network access control systems need to be flexible in controlling access for vulnerable systems. If patches are not available or cannot be installed properly (e.g., due to the inability of the owner or software dependencies), the machine is rendered useless until a fix is developed and properly tested, which can take several weeks [7]. This is further complicated in a publicly available mobile network where clients are not under the direct control of the network administrator. It is too simplistic to assume that disconnecting vulnerable systems constitutes a viable solution. However, **networking technology has sufficiently advanced, making more appropriate threat responses possible.**

The proposed quarantine approach is based on standard IP routing which eliminates the need for resource intensive network monitoring required by fingerprint-based systems. This allows the network

[1]Perfigo, which has been recently renamed Cisco Clean Access (`http://www.cisco.com/go/cca`), also has the ability to associate machines with roles and mark packets, resulting in an advanced system that can manage secure network access; although these features were not publicly documented at the time of this publication [11]. The authors thank Rajesh Nair of Cisco Systems for his comments and information concerning this product.

to run at full speed without the overhead of an intrusion detection system (IDS) or a fingerprint-based firewall, although these components can be integrated into the security authentication paradigm. The proposed system is also generic in that systems are quarantined for specific vulnerabilities, not specific worms, so new worm variants that exploit the same vulnerability are automatically thwarted. Since there is often a space of weeks between the patch for a vulnerability and the time a worm is released to exploit the same vulnerability [7], the proposed system provides protection before the malware is likely to exist. Furthermore, the proposed model does not require any client side tools, therefore it is effective for any client in the network. For example, the system does not require personal (host) firewalls or IDS software, which are not feasible to centrally manage in a publicly available mobile network. As a result, the system is able to successfully defend against internal malware threats.

The remainder of this paper is structured as follows. Section 2 discusses the issues of malware defense in a mobile environment, which is more difficult than in traditional fixed or static networks. The components of the proposed system, the concept of security authentication, and how the components interact are introduced in section 3. Section 4 discusses the system implementation for TCP/IP networks and provides a detailed description of the system function. Section 5 reviews the results of a proof-of-concept network that indicates current technology is sufficient to implement the malware defense. Finally section 6 reviews the proposed malware defense system and discusses future research.

2 The Failure of Malware Defenses in Mobile Networks

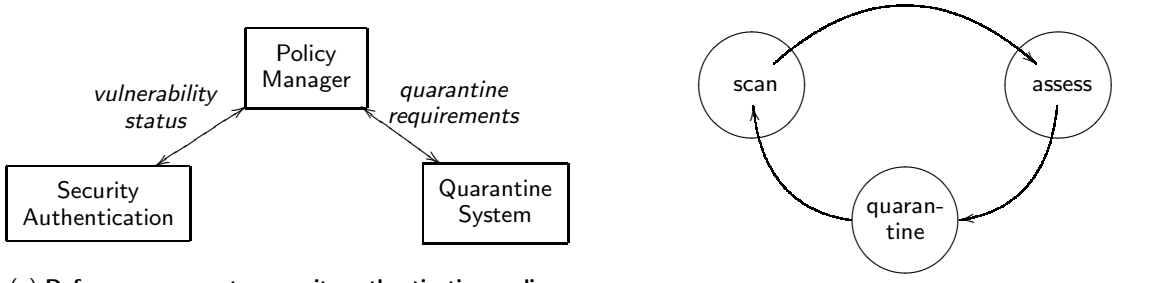
As the proliferation of mobile networks and ubiquitous computing occurs, the traditional inside and outside paradigm used to categorize threats is proving to be ineffective. In this environment, attacks from malware can start inside the secure network through malicious or simply naive agents. This is particularly the case with publicly accessible networks such as libraries, coffee shops, and universities where users bring their own machines into a network. Client machines in this environment are not under control of the network administrator and thus software may be unpatched and out of date. As a result traditional external firewall defenses are bypassed [6, 13, 18].

These mobile clients, however, are not only at risk to be infected, but are also a liability in that an infected client could consume significant network resources as it tries to propagate the worm, which adversely affects even the controlled clients. Even if the local network is monitored by a fingerprint-based system, a mobile client can connect to the network for a duration of time that is long enough propagate malware, but not long enough for current adaptive signature-based systems to react and disconnect the system from the network [18]. Unfortunately, personal (host) firewalls do not offer a realistic solution. Publicly available mobile networks will consist of machines with various operating systems and platforms. Given this heterogeneous and dynamic environment, the administrator has no direct control of client machines and is therefore unable to know whether the local policy of a machine is compliant with the overall policy.

System administrators have implemented wireless security tools and authentication mechanisms such as LEAP [12], to combat the possibility of guest machines disrupting a network. These are often employed to provide security via access control [2]. Unfortunately, this type of user or machine authentication falls short as a tool to prevent the inside spread of malware. It is common for individuals to access more than one network with a mobile computer. Even if a user authenticates correctly and is using the same machine that had been used in the past, it is possible that the client was on a completely insecure network elsewhere and has been infected by worms and other malware. Most networks are not structured in such a way to prevent internal hosts from compromising other internal hosts. Furthermore, often local communication is not monitored by an intrusion detection system because intrusion detection and packet filtering based on packet content are resource intensive. Therefore, the next generation of malware defenses must authenticate the user and the machine security.

3 A New Adaptive Defense for Malware

As described in the introduction, preventing worm propagation is quite a challenge even if all details of the worm are known before the worm is ever



(a) **Defense components: security authentication, policy manager, and quarantine system.**

(b) **Policy manager tasks: scan, assess, and quarantine.**

Figure 1. Malware defense system components and tasks.

released. Fingerprint-based systems such as Snort [14] are resource intensive and as such are difficult to implement over high speed connections without significant performance reduction. These signature-based systems have another weakness in that signatures are impossible to discover before specific malware is released and active.

Furthermore, it is difficult to effectively implement current malware defenses because of the new challenges associated with mobile networks. In this environment, threats can originate from any point in the network, not just the external world. A traditional firewall can protect from outside threats; however, the deployment of personal firewalls in a heterogeneous and mobile environment to defend against internal attacks is not possible, since machines may not be under control of the network administrator.

This paper introduces a malware defense system designed to contain vulnerable and infected machines in a new fashion. The strategy of the proposed architecture is to isolate systems based on the system vulnerabilities before they can become infected or attack others. This results in a defense against internal and external malware threats. As seen in figure 1(a), the proposed architecture is composed of three fundamental parts: a system to detect vulnerabilities, a system to enforce the quarantine, and a system to integrate and manage the overall security policy. These three parts must seamlessly work together to provide protection from attacks and are described in detail in the following sections.

3.1 Security Authentication and Vulnerability Detection

As discussed in the introduction, access to network resources is traditionally based on user authentication [2]. The primary objective of authentication is to bind an identity to a subject [2]. When applied to network access, authentication protects information and resources that are restricted to certain individuals. User authentication, however, provides no protection against malware. In a mobile environment, even individuals that should have access to certain network resources could use machines that have been infected from another source and are inherently insecure. Therefore, the proposed *security authentication* is fundamentally different from user authentication because it authenticates the security of the machine by detecting and characterizing the system vulnerabilities.

The system must be able to detect vulnerabilities remotely because not every client is under the control of the network administrator. Much like a unseaworthy boat, a vulnerable system is not fit for full network access. It is a weak point in the network which puts the host and the entire network at risk. Security authentication is a needed addition to user authentication to assess and quantify the risk of a particular system. As previously de-

scribed, not all insecure systems pose the same level of risk thus should be managed differently. The results of the vulnerability detection, or the security authentication credentials, are passed to the policy manager, discussed in section 3.3, which determines the appropriate action.

In terms of the authentication process, when a machine connects to the network, the security scanner initially probes all client ports for running services. Based on the results of the initial probe, it attempts to determine what services are running. Then the scanner tries to exploit known vulnerabilities of each service in an attempt to test the overall system security. Ideally the vulnerability detector would be akin to a *master worm* without a payload. This tool would attempt to exploit known vulnerabilities but not actually harm the system, and finally would report its analysis regarding the system security to the policy manager. The security authentication process occurs periodically to maintain the correctness of the vulnerability assessment.

3.2 Quarantine System

The quarantine system component has the responsibility of isolating a machine so it cannot become infected, infect, or attack any network hosts. However as previously described, the system should provide network connectivity commensurate with the security authentication level. The ability to provide of multiple levels of containment is different from other defense systems, that can only connect or disconnect machines.

Based on the perceived threat, which is determined via the security authentication process, the machine is given a certain amount and level of access. As previously described, access is restricted such that the machine cannot become infected, infect, or attack other hosts. However, enough network access is given to allow other programs to function properly. Machines can operate in a controlled fashion until a fix is developed and properly tested, which may require several weeks [7]. Quarantine can also be used to safeguard the defense system components and to assure the security of control information. Another desired feature is immediate protection, for example a host should be protected by default when it first comes onto the network and is later put into a less restricted position if it is secure. Finally, the system can protect against multi-headed malware by applying a more restrictive quarantine to the client.

To provide the desired quarantine functionality, the proposed system must integrate with standard networking technologies, topologies, and techniques. Isolating a system at the network layer (OSI layer 3), for example, prevents the propagation across interconnected LAN's. This is critical since the majority of worms employ a network address scan to find potential hosts [3, 7]. For example, very restrictive IP netmasks can provide a network layer *security cell*, as seen in figures 2 and 4. This cell ensures the quarantined system will not contact nor be contacted by any other clients without going through the default router or gateway. The router, acting as a packet filter, can then enforce traffic rules to control certain traffic and bandwidth usage.

Although network layer security cells provide significant protection, it is important to realize that clients are still connected to the same physical network. Consider the logical segmentation depicted in figure 2. Despite the segmentation at the network layer, spurious ARP requests and other traffic can be seen by all clients on the same switch. Thus an additional component of the security cell is needed to segment the network at MAC layer (OSI layer 2) [15]. Separation at the MAC layer prevents direct contact between system connected to the same physical network. Isolating systems at the network and MAC layers creates a proper security cell, where quarantined systems are truly limited in their network access.

3.3 Policy Manager

Although methods for detecting vulnerabilities, obtaining security authentication credentials, and quarantining systems have been discussed, an entity is needed to associate this information to the appropriate type and amount of network access. As seen in figure 1, the policy manager communicates with the other two components (security authentication and quarantine systems) and continually performs three critical tasks (scan, assess, and quarantine).

Once a machine enters the network it is initially placed in a restrictive security cell, where it undergoes security authentication (scan task). The policy manager reviews the results (assess task) and then places the machine in an appropriate security group (quarantine task). The tasks occurs periodically, giving machines the opportunity to move between

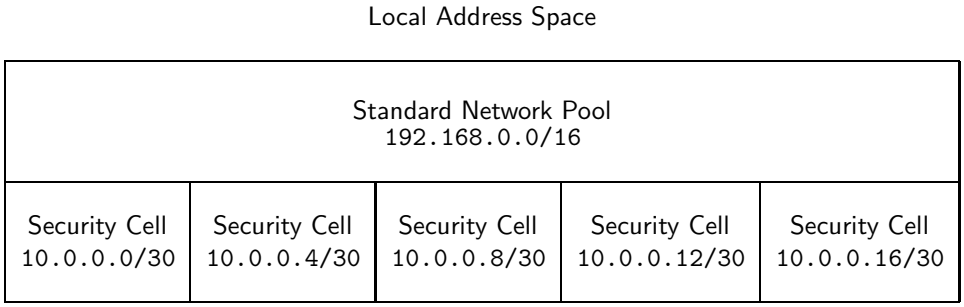


Figure 2. Example logical network segmentation with a general pool and network security cells.

groups for example after a software patch has been properly applied. A re-assessment can also occur if suspicious activity is detected (via intrusion detection systems [14], honeypots, honeynets, etc...). Regardless of why or when the tasks are performed, the objective is to place the machine in the correct security group.

Consider the following scenario: a system enters the network with an out-of-date version of the Apache httpd service running that has a vulnerability that allows remote arbitrary code execution. This information is discovered by the security scanner and passed on to the policy manager. Using this information, the policy manager can deduce that the client is in one of two possible states: vulnerable and infected, or vulnerable and clean. If the client is in an infected state, it is a hazard to the entire network. If the client is in a clean state, however, it is not dangerous, but merely at risk. It is difficult to distinguish between a vulnerable client and an infected client that is still vulnerable since a worm does not usually fix the vulnerability that it exploits. With the current tools, the systems are indistinguishable from a simple scan, however, these two classes of systems could be distinguishable with more sophisticated tools. The policy manager must determine an appropriate quarantine based on the specificity of the scan results and the security policy that is in place.

A simple policy would only offer two types of access, full or very restricted access. This type of policy protects any vulnerable system from further infection by restricting its access solely to update servers from which the system can be patched. Although this simple policy only offers two basic types of connectivity, it still better than current systems since it allows infected machines to access select network resources. This policy was utilized by the proof-of-concept system described in section 4, which can compensate for low specificity of information returned from a simplistic security scanner.

The second type of policy offers more controlled access by segmenting the network based on security groups as seen in figure 3. In this type of policy there exists a population of secure clients, a population of infected clients, and a population of known vulnerable but not infected clients. Utilizing security groups, systems are segmented from each other based on vulnerabilities. For example in figure 3, all clients are being protected from Apache worms while simultaneously clients vulnerable to Windows File Sharing worms are being protected from attack. In this mode the policy manager would notify the quarantine system to deny certain types of traffic that could spread the worms or compromise the vulnerable systems. Therefore, unlike the previous policy model (disconnecting vulnerable machines), this model allows some programs operate normally and securely even if vulnerabilities are present. This is beneficial considering the amount of time required to create and test software fixes.

A third type of policy would combine security authentication with user authentication to produce a hybrid system of security levels. This system would segment the network based on the type of services that exist in the organization, such as financial services, SQL services, WWW services, etc. Each client would employ user authentication to gain ac-

cess to a level and security authentication to show that the machine that is in use is safe to enter this level.

This section has described three different policy options; however, new policies as well as combinations of policies are also possible. The system is only limited by the accuracy of the security scanner and the complexity of the policy manager. Furthermore, this example only considered one vulnerability. However a security group can provide isolation for multiple vulnerabilities, thus defending against multi-headed malware.

3.4 Scalability

Although the proposed malware defense is described in terms of having one machine per system component, multiple security scanners and quarantine system agents can be utilized in a distributed fashion. The policy manager still coordinates access for the entire system and could perform load balancing to ensure that certain components are not overworked. Regardless of additional resources necessary for a large implementation, the network is able to run at full speed, which is in sharp contrast to fingerprint-based defenses. Therefore with the addition of a more advanced policy manager, the proposed system is scalable to different sizes of networks.

4 System Implementation

The previous section described a new security system that utilizes security authentication to defend against malware. Using this architecture, machines are authenticated based on system vulnerabilities and then isolated if necessary to prevent the spread of malware. The system consists of three components: security authentication, policy management, and system quarantine. While these system components have been described in general terms, this section discusses how they are implemented in a TCP/IP network.

4.1 Vulnerability Detector

Security authentication provides an evaluation of the vulnerabilities associated with a machine. It is important to obtain the most accurate and detailed information possible in order for the policy manager to determine the most effective quarantine.

Of the current available tools, Nessus offers the most advanced scanning functionality [8]. Nessus has the capability of remotely scanning a client to determine running services, the versions, and if the client is susceptible to specific security threats. The assessment library associated with Nessus is very comprehensive, covering a large variety of architectures, operating systems, and services. In contrast, Nmap provides a faster assessment of running services and versions [9], such as OpenSSH [10] and Apache [1]. These scanners can be used together to create a fast and comprehensive authentication system. For example, the results of an initial Nmap scan can be used by Nessus to conduct a more directed and thorough assessment. After the assessment, a machine with a known vulnerable version of any service is flagged as insecure. This information, security authentication credentials, is forwarded to the policy manager which can determine the appropriate action based on threat level and policy scheme.

4.2 Quarantine System

The quarantine system is responsible for restricting the network connectivity of groups of machines. Isolation is done to prevent the spread and attack of malware to other systems. Standard networking tools should be utilized, since it would not require clients to have any custom or specific software. As previously described, quarantining must be done at the network layer (OSI layer 3) and the MAC layer (OSI layer 2) to effectively defend against malware.

The Internet Protocol provides logical address segmentations (subnets), that form the basis for the network layer quarantine. For example the security cells shown in figure 2 can be easily created using subnets. Figure 4 depicts one IP security cell, where the netmask `255.255.255.255.252` represents a extremely limited subnet. There are two usable addresses in the cell, `10.0.0.1` and `10.0.0.2`. The security scanner and gateway occupies `10.0.0.1` and the client has the `10.0.0.2` address. A machine infected with a worm can only successfully scan one address (which is the security scanner itself) without passing through the default router or gateway. The security scanner is assumed to be secured by the network administrator and thus is not at risk of attack. All other traffic from the infected machine is directed by another important component of the quarantine system, the packet-filter/router.

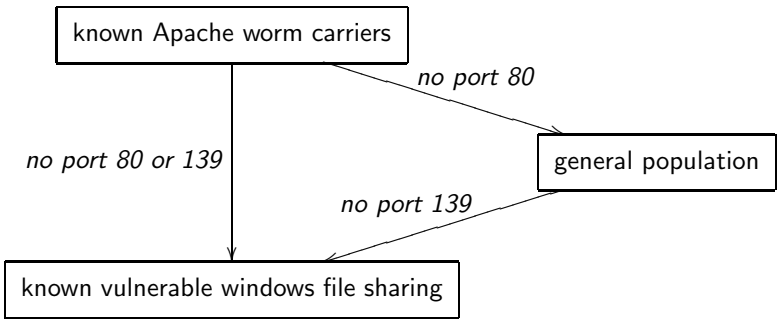


Figure 3. Example security groups.

IP Security Cell	
Netmask	`255.255.255.252`
Gateway	`10.0.0.1`
Network Address	`10.0.0.0`
Security Scanner	`10.0.0.1`
Client Address	`10.0.0.2`
Broadcast Address	`10.0.0.3`

Figure 4. Example IP settings for a security cell.

The quarantine packet-filter/router denies unwanted traffic between security cells and groups and facilitates communication between security cells that require interconnectivity. The specific behavior of the system is determined by the policy manager but enforced by the quarantine system. This functionality can be provided using `iptables` [19]. This can also be accomplished through advanced routing as long as the security policy scheme does not require port filtering, etc. Table 1 shows a sample configuration for a firewall, which reflects a simplest security policy. In this example, the general network occupies the `192.168.0.0/16` address space and the security cells occupy the `10.0.0.0/8` address space. These simple rules prevent communication between the security cells and the general network, and between security cells themselves. This prevents any machines in quarantine from being infected or from mounting an attack on other machines.

For MAC layer quarantining, a Virtual LAN (VLAN) can be used to separate machines connected to the same physical network, thus providing the appearance and functionality of multiple physical LAN's [15]. Using this approach, the VLAN boundaries would be aligned with the boundaries of the security cells and network providing layer 2 protection to supplement the aforementioned layer 3 protection. Layer 2 protection through VLAN's is a key addition to the quarantine system and is increasingly supported by most wired LAN's. Wireless LAN's can provide this functionality if the Access Point (AP) is equipped with Point Coordination Function (PCF) [15]. In this case, the AP could apply the MAC security rules to the arriving MAC frames, isolating the MAC traffic from different groups. Malware containment for wireless networks that do not rely on an AP for communication (e.g. ad-hoc networks) is a difficult problem and is the subject of continued research [17].

4.3 Distributing the Quarantine Information to Machines

The quarantine policy, which consists of MAC and network quarantine directives, must be distributed to the clients. The Dynamic Host Configuration Protocol (DHCP) is the basic tool for the system because clients can be configured with network parameters remotely [15]. DHCP allows

	Source		Destination		
Proto.	**IP**	**Port**	**IP**	**Port**	**Action**
*	10.0.0.0/8	*	10.0.0.0/8	*	drop
*	10.0.0.0/8	*	192.168.0.0/16	*	drop
*	192.168.0.0/16	*	10.0.0.0/8	*	drop
*	*	*	*	*	accept

Table 1. Example security policy for a network layer security cell.

remote specification an IP address, netmask, and lease renegotiation parameters. The use of DHCP allows host isolation to a security cell when it first enters the network. This accomplishes the goal of immediate protection.

For example, consider a DHCP server configured to model the network as shown in figure 2, where some cells have been eliminated for simplicity. When a new client performs a DHCP request, the client is issued an address from pre-configured security cells with a restrictive `255.255.255.252` netmask and a very short DHCP lease time. If the client has vulnerabilities, the DHCP server renews its address in a security cell until it becomes secure. If the client is secure, it is given an address from the standard network pool of addresses.

A sample DHCP configuration can be seen in figure 5. This sample configuration shows a shared physical network in which there are two separate subnets, `10.0.0.0/8` and `192.168.0.0/16`. The lease times for the `10.0.0.0/8` subnet are 60 seconds to facilitate a quick renewal after a security scan. The lease times on the `192.168.0.0/16` subnet are longer, 10 to 20 minutes, but still short to mitigate the threat of quickly developed malware. The first pool described is the secure pool and only known clients, clients that have passed a security scan, may receive addresses from this pool. The second pool described is a security cell with a very restrictive netmask which models a security cell as shown in figure 4. A standard configuration would have one additional pool to define each additional security cell, but these have been omitted from the configuration file sample for simplicity.

Unfortunately there are limitations with current DHCP implementations [16]. Once a client has received its address lease, there is no way to force the client to accept a different address. The address change can only occur if the client requests a lease renewal. Hence, if a client is found to be insecure in the middle of its standard pool DHCP lease, the system is unable to logically relocate the client into a security cell until the client requests a lease renewal. During this period of time, a significant number of hosts could be found to have a new vulnerability and become infected. This is not a limitation specific to this security mechanism, however. RFC 3203 [16] calls for a DHCP reconfigure extension in which a DCHP server can send a FORCERENEW message to a client to force an immediate lease renegotiation. This would provide a solution to this issue, but this problem is currently mitigated by a choosing a short DHCP lease time that ensures that most clients would renew in the time between when a vulnerability is discovered and a worm is crafted to exploit the vulnerability. Furthermore, another workaround is available at layer 2 in that the machine could be removed from its current VLAN until it requests a new address. This is an extreme measure, however, and considering past worms, a lease time of up to two weeks would be acceptable, but a time of one day would avert all but the fastest attacks.

4.4 Policy Manager

As previously described, the policy manager is responsible for mapping the security authentication credentials to a particular level of access for each security cell. The policy is determined in advance by the system administrator can be a simple scheme separating vulnerable machines from others, or a complex system of security cells. Again, this is dependent on the level of detailed offered by the security authentication system and the needs of the network.

Once a machine has connected to the network and undergone the security authentication, the pol-

```
shared-network example {
subnet 10.0.0.0 netmask 255.0.0.0 {
    default-lease-time 60;
    max-lease-time 60; }
subnet 192.168.0.0 netmask 255.255.0.0 {
    option routers 192.168.0.1;
    default-lease-time 600;
    max-lease-time 1200; }
pool {
    allow known clients;
    range 192.168.0.10 192.168.0.50; }
pool {
    deny known clients;
    range 10.0.0.2;
    option routers 10.0.0.1;
    option broadcast-address 10.0.0.3;
    option subnet-mask 255.255.255.252; } }
```

Figure 5. Example dhcpd configuration.

icy manager receives security authentication credentials. The policy manager, implemented for example as a daemon process, maps the credential to the appropriate security cell. After determining how the security policy applies to a client, the policy manager sends the specific quarantine and route information to the quarantine system. However, the policy manager can also be independent of the network technology. In this case, the policy manager only needs to inform the quarantine system of the machine identity and the appropriate security cell. The quarantine system can then invoke the appropriate network and MAC layer functions.

5 Experiment Results

A proof-of-concept system was developed to test the merits of the proposed malware defense in a mobile network environment, as well as the suitability of current networking technology. As seen in figure 6, the system consisted of a mobile network where four computers were interconnected via a 1 Gbps switch. Each computer, installed with Gentoo Linux 2004.1 [4] (2.6.7 kernel), served as either a mobile client or the malware defense system.

Machine A implemented the proposed malware defense system consisting of the security scanner, policy manager, and the quarantine system. Nmap was utilized for the security scanner, while IP Forwarding and IP Tables were utilized for quarantining [19]. As previously described, Nmap has the ability to scan for open services and, in certain circumstances, identify service versions. IP forwarding and IP tables provide routing and filtering support required for isolating machines in certain security cells. A daemon process was created for the policy manager, which mapped the vulnerability status of a mobile to the appropriate security cell.

The mapping process utilized a simple file that described the defense policy. As described in section 3.4, the policy implemented separated machines two basic groups, vulnerable and secure. Note, when a machine enters the network, it is automatically placed into a security cell and is assumed to be vulnerable until the security authentication determines whether to continue quarantine or allow the machine onto the network. The secure portion of the mobile network consisted of the `192.168.0.0/16` subnet, while the security cells were constructed on the `10.0.0.0/8` subnet as shown in figure 2. Although the security cells are logically close and share

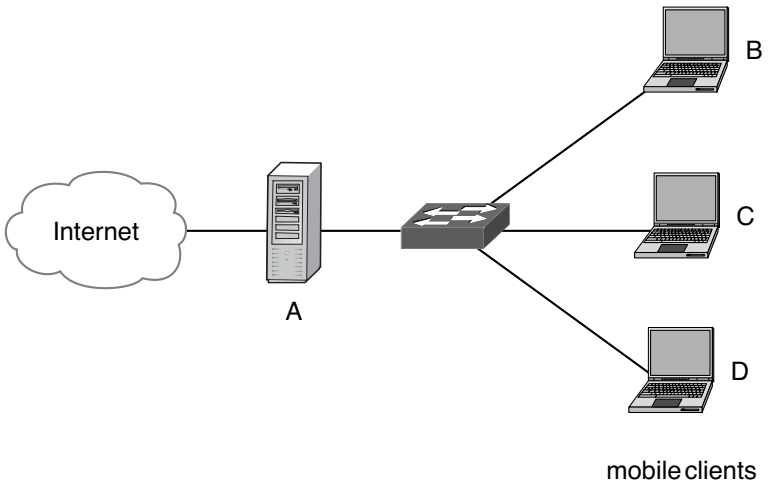


Figure 6. The network configuration used for system testing. Machine A implements the proposed defense system, while machines B, C, and D are mobile clients.

the same address space, the security cells are strictly separate at the network level and have no interconnectivity. Three security cells were constructed as quarantine areas.

Figure 7 shows the basic operation of the complete system at a high level. A new mobile client enters the network and is placed into state 1, an initial security cell. After a short period of time, the security scanner scans the client to discover any known vulnerabilities. This is represented as state 2 in figure 7. After the scan is complete, the security scanner relays the security authentication credentials to the policy manager. The policy manager then decides what kind of access to grant the client. If the client has no known vulnerabilities, the client is given an address from the standard network address pool and thus rests in state 3. If this is not the case, the machine returns to a state 1, the initial security cell, and the process can repeat if necessary. The security cell has access to either a local update server or the Internet so the system administrator of that particular system can update the system when possible to gain additional network connectivity. The short lease time ensures that clients are admitted to the normal network shortly after its security authentication is complete.

To represent different vulnerabilities, the mobile clients (machines B, C, and D) executed different versions of OpenSSH [10]. Again, machine A acted as the security scanner, policy manager, and the quarantine system. Client C had an older version that was known to be insecure, while client B had a current version of OpenSSH, and client D had no services running. Once a client entered the network, it was assigned a security cell address via DHCP. Afterwards, it was then scanned by machine A for known vulnerabilities. Machine B, with a current version of SSH, passed the security scan and was marked as such in the DHCP configuration file. Upon DHCP renewal which occurred within one minute, it was then assigned an address from the `192.168.0.0/16` pool and it immediately moved to the new network where it could access other network services. Client C, with an insecure version of OpenSSH, was also assigned a security cell address via DHCP. During its scan by machine A, however, it was noted to be running this insecure version and it was not placed into a trusted clients section for DHCP. Upon DHCP renewal, client C again received an address for a security cell and was denied access to the standard network. It is important to note that this client was not simply disconnected from the network, but maintained limited access to select resources, which would allow the user to patch this machine.

For stress testing, machines were scripted to turn vulnerable services on and off for several minutes at a time. When the client was in a secure state, it was given an address in the `192.168.0.0/16` subnet, but upon DHCP renewal, if the client had

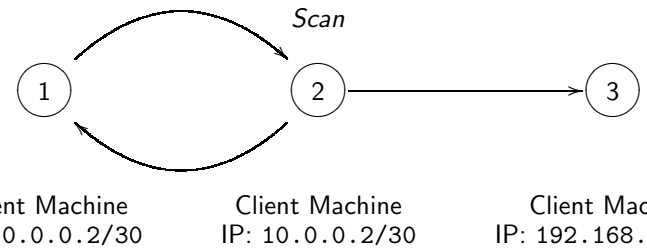


Figure 7. Three possible client states for the example malware defense.

reached an insecure state, it was relegated to the security cells until it again became secure. Machine *A* seamlessly moved the clients from security cell to the standard pool and vice versa. The system was tested for several days and successfully defended the network without any problems. Therefore, this example demonstrates the proposed malware defense is able to successfully manage and quarantine machines based on vulnerabilities using current networking tools.

6 Summary and Conclusions

Effective malware defense is a difficult and increasingly important issue for computer networks; however, current defenses are often unable to manage these threats. Current solutions rely on malware fingerprints (signature) to be known a priori, which is not always possible. In certain instances, these systems also require large amounts of processing, for example data mining capabilities, that must be done in real-time to be effective. Furthermore these systems are not suitable for a mobile environment, where the gateway firewall is easily bypassed and attack occur from the inside. The deployment of personal firewalls does not offer a realistic solution since machines in a publicly available mobile network are not under the control of the network administrator.

This paper introduced a new malware defense system consisting of three basic components: security authentication, quarantine system, and the policy manager. Security authentication is an effective and anonymous method to ensure the safety of hosts on a network. In contrast to user authentication, security authentication detects and characterizes the vulnerabilities of the machine in question. This new type of authentication is necessary since an authenticated user can bring a vulnerable or infected machine into a secure network. Therefore, the system is particularly effective at preventing the spread of malware inside a local network (e.g. mobile environment) where traditional firewall systems are no longer effective. Furthermore, the proposed defense is not restricted to a certain instance of malware since vulnerabilites are targeted instead of fingerprints.

Security authentication credentials are used by the policy manager to quarantine the machine. The quarantine system isolates the machine by placing them in a security cell, which utilizes the network and MAC layers to prevent the machine from being infected or attacked by other hosts and vice versa. Unlike current systems that disconnect a suspect machine, the quarantine system affords the machine a certain level of network connectivity. This allows the machine to still function until the malware or vulnerability is addressed. This paper also discussed how the proposed system can be applied to TCP/IP networks utilizing current network technology and tools. Advanced routing and VLAN's offer the necessary quarantine abilities, while Nessus and Nmap are sufficient for simple security authentication. The proposed malware defense was successfully implemented and tested using these tools and basic Linux equipped computers.

Although the proposed system provides the basis for better malware defense, areas for future research and improvement exist. Additional research is needed to better define the security policy employed by the policy manager. The policy describes how the manager maps security authentication credentials to security cells and how the cells should interact. Three different types of policies were introduced to demonstrate the variety of possible policies (simple to complex); however, malware defense

policies could benefit from a more formal definition. More research is also needed to determine how the system can properly scale to larger networks. As discussed in this paper, it is possible to distribute the proposed system across a large network by replicating the components; however, an additional management system is needed to integrate these smaller pieces. Finally, more research is necessary to determine the proper amount of time between successive scan-assess-quarantine operations. The defense system should update the security cells as soon as possible, while minimizing the overhead associated with the operations. This can be addressed by integrating intrusion detection technology with the defense system, causing the system to be more reactive to suspicious behavior.

References

[1] Apache HTTP Server Project. `httpd.apache.org`.

[2] M. Bishop. *Computer Security: Art Science*. Addison Wesley, 2003.

[3] D. Ellis. Worm anatomy and model. In *Proceedings of the First ACM Workshop on Rapid Malware*, 2003.

[4] Gentoo Linux. `www.gentoo.org`.

[5] G. V. Hulme. Under attack. *InformationWeek*, July 2004.

[6] J. C. Hung, K.-C. Lin, N. H. Lin, and L. H. Lin. A behavior-based anti-worm system. In *Proceedings of the Seventh International Conference on Advanced Information Networking and Applications*, 2003.

[7] D. M. Kienzle and M. C. Elder. Recent worms: A survey and trends. In *Proceedings of the First ACM Workshop on Rapid Malware*, 2003.

[8] Nessus Security Scanner. `www.nessus.org`.

[9] Nmap Security Scanner. `www.insecure.org`.

[10] OpenSSH, the Open Source Version of the SSH Protocol. `www.openssh.org`.

[11] Perfigo. *Neutralizing Internet-borne Threats at the Network Edge*. `www.perfigo.com`.

[12] L. L. Peterson and B. S. Davie. *Computer Networks: A Systems Approach*. Morgan Kaufmann, second edition, 2000.

[13] S. Sidiroglou and A. D. Keromytis. A network worm vaccine architecture. In *Proceedings of the Twelfth IEEE International Workshop on Enabling Technologies*, 2003.

[14] Snort, Open Source Network Intrusion Detection System. `www.insecure.org`.

[15] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, fourth edition, 2003.

[16] Y. T'Joens, C. Hublet, and P. D. Schijver. DHCP reconfigure extension. IETF RFC 3203, December 2001.

[17] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang. Security in mobile ad hoc networks: Challenges and solutions. *IEEE Wireless Communications*, pages 38 – 47, February 2004.

[18] C. C. Zou, L. Gao, W. Gong, and D. Towsley. Monitoring and early warning for internet worms. In *Proceedings of the First ACM Workshop on Rapid Malware*, 2003.

[19] E. D. Zwicky, S. Cooper, and D. B. Chapman. *Building Internet Firewalls*. O'Reilly, 2000.