Worms vs. Perimeters: The Case for Hard-LANs

Nicholas Weaver
ICSI
nweaver@icsi.berkeley.edu

Dan Ellis
The MITRE Corporation
ellisd@mitre.org

Stuart Staniford
Nevis Networks
stuart@nevisnetworks.com

Vern Paxson
ICSI
vern@icir.org

***Abstract*— Network worms—self-propagating network programs—represent a substantial threat to our network infrastructure. Due to the propagation speed of worms, reactive defenses need to be automatic. It is important to understand where and how these defenses need to fit in the network so that they cannot be easily evaded. As there are several mechanisms malcode authors can use to bypass existing perimeter-centric defenses, this position paper argues that substantial defenses will need to be embedded in the local area network, thus creating "Hard-LANs" designed to detect and respond to worm infections. When compared with conventional network intrusion detection systems (NIDSs), we believe that Hard-LAN devices will need to have two orders of magnitude better cost/performance, and at least two orders of magnitude better accuracy, resulting in substantial design challenges.**

I. Introduction

Network worms—self-propagating network programs—represent a substantial threat due to four distinct worm properties: speed (worms spread faster than human reactions), ability to penetrate networks (worms spread widely), democratic nature (just about anyone can use a toolkit worm), and arbitrary payload (worms can carry whatever code an attacker desires). Each of these properties has implications when designing worm defenses.

It has now been well established by both theoretical modeling [9], [15] and actual worms [7] that, due to their extreme speed, any reactive portion of the defense which seeks to detect, analyze, and respond to a worm must be automatic. But where should these defenses be placed in the network?

Architectural decisions about where to place worm defenses in a network need to take into consideration how worms have historically crossed many perimeters. Defenses should be deployed where worms cannot trivially bypass them, while being composable so that a failure of one perimeter has minimal consequences.

In the rest of this paper, we provide a working definition of a perimeter, and how perimeters form a critical component of worm defense. We then review and elaborate on several strategies worms have used to cross coarse-grained perimeters, such as firewalls [4] and even "air gaps." We also review the techniques malcode has used to attack anti-virus and other fine-grained perimeters. In short, contemporary architectures (consisting of only very coarse-grained and very fine-grained perimeters) are inadequate with respect to the worm problem.

The position we develop from the arguments we present is that substantial portions of any robust worm-defense will need to be deployed *throughout* the local area network. This will require the development of "Hard-LAN" devices: network components designed to both detect and respond to worm infections throughout the local area networks of enterprise networks.

Although we argue that this is a necessary step, creating Hard-LANs will encompass substantial research challenges, for not only is it currently difficult to detect many classes of worms, but Hard-LAN defenses will have to operate at substantially higher data rates, at substantially lower prices, when compared with conventional intrusion detection systems. We estimate that, due to an order of magnitude higher data rates, combined with an order of magnitude more devices in an institution, Hard-LAN defenses will need to provide a cost to performance ratio of two orders of magnitude better than contemporary network-based intrusion detection systems (NIDSs) [11], [12], if system cost is to remain comparable with current defenses. Likewise, the accuracy, measured in expected false positives per bit of traffic observed, will also need to be improved by at least two orders of magnitude.

II. Perimeters and Postures

Architectural decisions about where to place worm defenses in a network need to account for how worms have historically defeated perimeters. We first define what we mean by both a *perimeter* and a *posture*. We next briefly discuss properties of perimeters and postures and provide a set of guiding principles for perimeter and posture evaluation. The following section then applies these principles to evaluating contemporary perimeters and postures.

A. Perimeter and Posture Definitions

A *perimeter* is a controlled barrier that implements a policy on communication between elements protected by ("inside") the perimeter and elements outside the perimeter. For our purposes, these elements represent network computers or hosts. Communication may move along many paths between elements. These paths of communication can take many forms, including network traffic, portable media, or even entire computers. A perimeter can only impose its policy on communication which crosses through the perimeter along checked data paths. Thus a severe limitation of all perimeters

Trend Micro, Inc.
EX1011-1

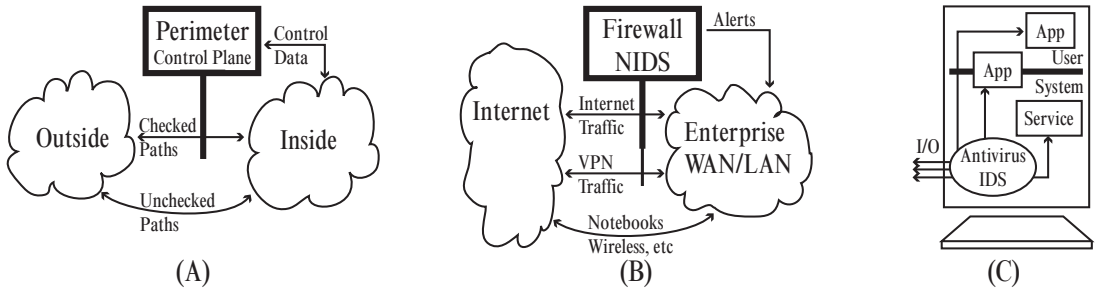

Fig. 1. (A) The conceptual view of a network perimeter. (B) Firewalls and NIDS as a coarse-grained network perimeter, monitoring the Internet traffic to a large institution. (C) End-host anti-virus viewed as a perimeter monitoring all I/O on the host system.

is that they can't observe communication which either goes around the perimeter on unchecked data paths or never leaves the perimeter.

A perimeter also has a control plane, which contains the logic that defines the perimeter's operation. This includes the logic to interpret (sensors) and modify (actuators) the constraints imposed upon traffic crossing the perimeter. This logic allows the perimeter to determine which traffic to allow, which traffic to modify, and which to interdict completely. The logic within the control plane may also allow the perimeter to communicate with other devices, as part of a larger defensive system. Figure 1(A) provides a depiction of a generalized perimeter.

An enterprise security *posture* consists of one or more perimeters and the properties that apply to perimeters, by aggregation, apply to enterprise postures. An enterprise posture is *robust* if the overall system security is not affected by the failure of an internal perimeter.

B. Perimeter and Posture Properties

The following properties describe a perimeter and, by aggregation, a posture. The *coverage* measures how many elements (hosts, in our case) are inside the perimeter. The *coverage* of a perimeter also affects the number of paths across the perimeter, while *completeness* is the fraction of the data that is checked verses unchecked. Thus, coverage refers to a (relatively) static property and completeness to a dynamic property, since it depends upon over which paths the data winds up flowing.

Practically speaking, no perimeter or posture is perfect. The real value of these properties is in helping network architects evaluate alternative perimeters to be included in an architecture. In cases where a perimeter is not fully complete, it may still be *adequately* complete, satisfying requirements of the architect for the given operational environment.

A perimeter *breach* is when explicitly malicious data flows across or around a perimeter. We must assume worst case consequences for a breach: nothing can be asserted about those elements that were protected by that perimeter alone, although the perimeter can still monitor subsequent traffic. Even worse is a breach of the control plane: when this occurs, the attacker has gained complete control of the perimeter.

C. Principles For Constructing Perimeters and Postures

We offer several principles as useful for evaluating the effectiveness of perimeters and postures:

- Perimeter control plane failure or perimeter breach results in the loss of the elements protected by only this perimeter.
- Perimeter control plane failure means that the perimeter can no longer provide egress protection (halting outbound attacks), and cannot provide reliable information as part of a larger defense.
- As the coverage of the perimeter increases, the likelihood of a perimeter breach never decreases and usually increases.
- As the coverage of the perimeter increases so does the cost of a breach.
- *Defense in depth* (an architectural design wherein a posture consists of conjoined perimeters that fail independently) and *containment* (restricting failures to a small portion of the network) minimizes the expected loss due to a breach.
- An accurate perimeter (which filters *only* "bad" data, so no false positives) is generally more valuable than a sensitive perimeter (which filters *all* "bad" data, but perhaps also some "good" data, so potential false positives). Layering accurate perimeters improves the posture, whereas layering sensitive (but inaccurate) perimeters usually reduces system reliability as the rate of false positives increase.

We use these guiding principles to evaluate contemporary perimeters and postures, and use them to formulate where worm defenses will need to be embedded in the network.

III. Contemporary Perimeters and Postures

Contemporary network postures generally consist of two perimeters: coarse-grained perimeters around large groups of machines and fine-grained perimeters running on end-systems. The enterprise network perimeter is the most commonly deployed coarse-grained perimeter. It consists of corporate firewalls on Internet service provider (ISP) boundaries, network-based intrusion detection systems (NIDS) deployed at the ISP boundary (Figure 1(B)), or a complete disconnection (air gap), isolating the network from the rest of the Internet.

The most common fine-grained perimeter runs on the end host, such as an end-host anti-virus solution and a personal firewall (Figure 1(C)). This software runs on the end-host it is attempting to protect, monitoring all I/O and other system state. Most enterprises employ a two-layer approach, combining end-host protections with a coarse network perimeter.

Enterprises may have additional perimeters in their posture, including physical access control and policies concerning mobile devices. These may or may not be relevant with respect to the worm attacks, depending on the vectors the worm employs and the time scales over which detection and response must occur.

A. The Enterprise Network Perimeter

The coarse-grained perimeter usually deployed consists of a firewall [4] at each ISP connection point, network-based intrusion detection system (NIDS) sensors, and a corporate email server anti-virus solution, as well as VPN concentrators for external access. Historically, this enterprise network perimeter has been adequate for many enterprises. Little data got on the network that did not pass through a checked path at the firewall and inspected by the NIDS sensors. The environment was such that the wired network was the primary path for data flow because of its relative ease and good bandwidth. The environment has changed, however, so that there are now numerous unchecked high-bandwidth data paths (Figure 1(B)). As a result of the changing environment, the enterprise network perimeter is not adequate for worm defense.

The enterprise network perimeter still has significant advantages and should not be abandoned. It offers significant coverage: all internal enterprise hosts receive some protection from it. It provides a good source of understanding some of the data that comes into and leaves the enterprise. However, the proliferation of unchecked data-paths results in inadequate completeness and accuracy, and worms have successfully exploited these flaws.

Even internal firewalls will often not stop these paths. Many institutions are deploying firewalls or network VLAN/ACL based access controls between large subgroups. Yet as these boundries are still very coarse, they have most of the same benefits and limitations as the conventional firewall.

The only checked paths across the enterprise network perimeter are those paths that enter through the network gateway at the ISP boundary. Whereas this was previously an adequately covered perimeter, other unchecked data paths have proliferated. The following paths include just some of the mechanisms worms have used to breach the enterprise network perimeter [13]: portable media (e.g., CDs, portable hard drives, USB pen drives, floppy disks), mobile computing devices (e.g., laptops, personal digital assistants [PDA]), wireless access points (WAP), virtual private networks (VPN) tunnels and modem access. We illustrate how some of these unchecked data paths have been used to breach the enterprise network perimeter.

Code Red II [3] spread only by attacking IIS web-servers listening to TCP port 80. Yet it infected some organizations that blocked inbound port 80 traffic. The vectors that allowed this were “trusted” organizations with VPN access; perimeters (if they exist) at VPN tunnel endpoints are usually much less constraining than the enterprise network perimeter.

Likewise, wireless access points commonly bypass the perimeter. Worse, these access points may appear spontaneously. Some laptops come with a network interface card (NIC) capable of serving as a access point, with the default installation enabling this service. For example, we know of one instance when such a laptop was connected to an enterprise network, automatically configured itself as an open access point, and silently bridged two networks. UDP packets containing the Slammer [7] worm came in via the bridge from a remote infected host, external to the institution, which had automatically associated with this spontaneously-generated open access point. (Note that the laptop itself was *not* infected, it was simply a “carrier,” in effect.)

It may prove difficult to even detect the appearance of such spontaneous access points within an enterprise, especially in geographic regions where access point density is high and the rogue APs use strong authentication. For example, a recent test in McLean, Virginia revealed over 300 access points accessible from a single point, using a small amplifying antenna. Enforcing an enterprise policy banning the deployment of access points connecting to the enterprise network is currently difficult even in areas where there are few access points, although there are some products in development to monitor and respond to new access points: detecting their arrival and then isolating them from the network by reconfiguring the LAN’s switches. The fact that wireless access points are easy to deploy and provide excellent functionality makes them common unchecked data paths into an enterprise network.

Laptops without wireless networking are still an unchecked data path. Many have observed that laptops have been taken into other domains (e.g., when a user takes the laptop home), infected in the remote domain, retained the worm while being taken back into the enterprise network (e.g., by putting the laptop in sleep or standby mode), been reanimated and infected other portions of the enterprise network. Mobile laptops represent one of the most common paths used by the Blaster [16] worm, as many enterprises had reconfigured blocks to stop all Windows RPC traffic at the firewall, yet were still infected by the worm. This unchecked data path has been used even by Slammer, which is only memory-resident.

Hybrid pathogens, such as worms that can spread as a virus as well (e.g., Nimda [2]), have also spread on media such as floppies, portable hard drives, and USB pen drives.

There are other unchecked data paths as well along which worms may be able to spread. The enumeration above, however, should demonstrate that the enterprise network perimeter is not adequately covered. Even when the network has an air-gap, these data paths are still relevant.

Not only are there many unchecked data paths, but these alternative data paths have significant capacity. The amount of data leaving or entering enterprise networks via these unchecked data paths is nearly impossible to estimate. Yet

current trends and observations of these alternative data paths being used by worms to spread support the conclusion that there is significant data flow through many of these and possibly other data paths.

Finally, the accuracy of the enterprise network perimeter is questionable at best, due to at least two factors. The first factor is that although a perimeter does check a particular data path, its policy may be out of date. The second mechanism is the lack of expressiveness in policy or signature languages.

The rapid spread exhibited by malcode (especially worms) outpaces signature distribution mechanisms. The signatures for both NIDS and email server anti-virus products need to be developed, made available, downloaded, and installed before they take effect. As the development happens after the malcode has already been released and can often take hours, it is possible for the malcode infection to peak before the signature is developed [15]. Nimda [2] is one example of malcode that spread widely before a signature was released for either its client-server attacks or its email footprint. No signature development and deployment mechanism that operates on a human time scale can compete with fast spreading worms.

Finally, it is clear that the techniques worms have employed to cross through and around perimeters could be enhanced by attackers. Rather than relying on blind luck, a worm copy running on a system equipped with a wireless NIC could scan the environment searching for open access points. A worm running on a laptop could be engineered to go dormant, only reawakening when plugged into a new network. Thus, we must assume that the coarse-grained perimeter will be penetrated by well-engineered worms. Since these perimeters encompass so many hosts, such failures are usually catastrophic.

B. The Host-Based Perimeter

Perimeters deployed on an end-host usually have the right level of visibility and access to be adequately covered, complete, and accurate. All data crossing a host must cross through the operating system, enabling the end-host perimeter to observe the I/O. As the logic for determining the appropriateness or maliciousness of any action is available to the end-host at either the operating system or application layer, the end-host can be as accurate as the logic permits.

However, host-based perimeters have two major limitations. First, for large enterprises they are exceedingly difficult to *manage*, due to sheer scale and the inevitable diversity in terms of end system configurations it brings. Second, such perimeters are *brittle*: if a worm manages to penetrate the perimeter, the defense mechanism loses its visibility into the worm's progression and its ability to contain the infection. This is because it is generally infeasible to maintain the integrity of the control plane after a breach.

Management is currently difficult but many have adapted, such as enterprises where end-host firewalls and antivirus mechanisms are mandated and deployed on all systems to provide an additional layer of protection. But this layer can only prevent an infection, not contain one. On current systems, once an attacker can escalate to system-level privileges, any egress protection is lost as the defensive perimeter and the control plane's own protections are the same.

It is already common for viruses to disable anti-virus software, while the egress protection of personal firewalls can be corrupted by Trojans overwriting the memory space of authorized programs. Escalating from limited access to super user access is often quite easy, as the vulnerabilities leveraged are usually not characterized as "'critical"' and often go unpatched. In other cases, including the Code Red II [3] and Blaster [16] worms, the exploit gains full system-level access directly, as the targeted service is already running with full system privileges.

TCPA [17] or related concepts such as virtual machines [18] might prevent a breach of the perimeter from breaching the end-host control plane, but actually constructing such systems remains a significant, open problem. This is especially difficult when grafting TCPA onto existing, legacy operating systems. Thus, at least for the near future, we must assume that the successful compromise of an end-host will compromise all defenses placed on the host, and the resulting brittleness makes host-based perimeters an inadequate defense for worms.

Finally, placing additional security layers may even increase the risk if an attacker can target the security software. The Witty worm [8], [19] infected ISS's end-host firewall software, overwriting the hard-drives of the systems which the firewall software was attempting to protect.

C. The Two-Layer Enterprise Posture

As no perimeter is perfect, layering perimiters provides increased coverage, completeness, and robustness. Although layering the above two perimeters provides increased protection over either one in isolation, the two-layer enterprise posture is not adequately robust. Given the porousness of the enterprise network perimeter, the composition only slightly improves on the brittleness of host-based perimeters, and does nothing to address manageability difficulties. Thus we believe that the two-layer enterprise posture is not adequately robust when attempting to defend networks against the impact of worms.

IV. Worm Defense Components

A comprehensive worm defense encompasses prevention, detection, analysis, response, tolerance, and recovery. Prevention represents the ability to resist an initial infection. Detection is the ability to discover that a new worm is operating within the network. Analysis seeks to monitor the progress of a worm and to find particular features of interest. Response devices change the network to resist an infection. Tolerance is the ability to withstand a given level of infection without disrupting normal operation, often by containing the infection to a small portion of the network. Recovery attempts to restore normal operation after an infection.

Perimeters play key roles in prevention, detection, response, and tolerance, and gather critical information required by analysis systems. Prevention is simply the obvious: if the initial configuration would stop the worm, the perimeter works.

Otherwise, it fails. Detection requires monitoring the traffic crossing the perimeter for signs of a worm, which is then fed into analysis systems. Response uses the results of analysis to change the perimeter's state intended to prevent further infections. Tolerance comes from a perimeter's ability to contain an internal infection. By placing enough containment-based perimeters through a network, it should hopefully be possible to limit a worm's damage.

Thus, we wish to deploy perimeters in locations where we can check as many data paths as possible, minimize the coverage of any particular perimeter, yet ensure that all assets are adequately covered. Also of concern is maintaining control-plane integrity, as any perimeter which cannot maintain its control system is largely limited to prevention and preventative-response roles, as once the worm breaches the perimeter's control plain, the perimeter fails completely.

Beyond their vulnerability to initial penetration, coarse-grained perimeters lack visibility into a worm's internal spread. Although an initial worm-detection may require only a single event, robust defenses will require continual monitoring to determine whether the current defensive posture is effective at halting the worm. Since the coarse-grained perimeter lacks this visibility, this becomes a significant limitation if these perimeters are to play a substantial role in worm defense, by feeding detection information into analysis systems.

Many worm defenses, such as worm containment [10], [14], [21], attempt to limit the damage an already infected host can inflict, tolerating a small amount of infection in exchange for halting the worm's overall progress. In particular, detection comes from observing patterns of network activity that manifest much more strongly (or only) in the presence of a local infection [10]. However, any defense which relies on detecting, tolerating, and responding dynamically must maintain control plane integrity even after the perimeter itself is breached.

Likewise, topological worms [20] such as the Morris worm [5], as well as contagion worms [15], are highly likely to penetrate firewalls. Both of these classes of worm use information about previous communication as a technique for finding new targets. Since they attack using information about either current or previously established relationships, most firewalls will allow a new topological or contagion worm to pass.

V. The Need for Hard-LANs

As a consequence of the failings of both coarse-grained perimeters and end-host protective mechanisms, we believe that more perimeters need to be layered throughout the enterprise network. Such layers need to be very tight, containing as few hosts as possible with the control plane in the network and not accessible to the user or operating system it protects.

By using separate devices in the LAN, we can maintain control plane integrity after infection. Thus LAN-based defenses can act to detect and contain an infection to a small section of the network, rather than just attempting to prevent an infection. Yet because the tightest perimeter only protects a few end-hosts near the edge of the LAN, these perimeters are both harder to penetrate and the results of a penetration are less catastrophic, aiding both prevention and tolerance.

Likewise, LAN devices have the best network visibility when monitoring end-hosts, contributing greatly to both detection and response. Since LAN-based devices can see most or all network traffic from both compromised and uncompromised hosts, they can hopefully detect a worm earlier on in its lifecycle, limiting the damage. And by blocking the traffic at the source, LAN-based devices can prevent a worm from contacting or compromising more systems.

Thus, if we wish to build networks which are robust to malice, we believe that the LANs will need to be hardened, either though the use of network cards outside of the host computer's control [1], devices located on the network links, or logic embedded in the switches themselves. In all cases, this pushes the defense to the edge of the LAN, close to the end-hosts, while still residing in the network. Only in this location can we hope to robustly resist a worm's attacks on both coarse-grained perimeters and end-host control planes, while preventing a compromise of our defenses from also immediately compromising the end hosts being protected.

This has significant implications which will affect both the design and implementation of worm defenses. Data rates in the LAN are substantially higher than those experienced by coarse-grained perimeters, yet the defenses themselves will need to be substantially cheaper, if we wish to retain the current NIDS model as a starting point.

The data-rates are just an unfortunate reality of the evolution of networking. While a 45 Mbps T3 connection Internet connection costs over $2,000/month, LAN bandwidth is vastly cheaper, with 24 port Gbps Ethernet switches selling for less than $1,500 and with 10 Gbps devices beginning to reach the market. Thus, most modern installation will use Gbps links for the aggregate system uplinks and many new installations bring Gbps all the way to the desktop.

While Gbps NIDS represents the current extreme of NIDS products, Hard LANs will regularly have to operate at Gbps rates. Protecting just a single, bidirectional Gbps link requires a device capable of processing 2 Gbps of data. A PC-based system would need to move 4 Gbps of data through the network cards if it was a pass-through, full-bandwidth device on a Gbps LAN. And if the defenses are integrated into the switch itself, aggregate bandwidth requirements can easily exceed 40 Gbps. This situation grows even worse when 10 Gbps links grow common.

Not only are data rates substantially higher, the cost of each perimeter will need to be substantially lower if we wish system cost to remain comparable to the current Firewall/NIDS approach. Rather than protecting an entire institution, each edge device in a hardened LAN only encompasses a few systems. Thus it will require many more devices to provide adequate coverage.

Combined together in bits per second per dollar, cost/performance for Hard-LAN edge devices will need to be two orders of magnitude better than current, coarse-grained perimeters if the final system cost is to be comparable. This

will require both new techniques and implementations, which will need to be co-developed to provide effective defenses. Thus, these perimeters will need router and switch-like specialized designs, able to support high data rates at low cost, rather than using conventional PCs and related general purpose devices.

Hard-LAN devices also need to be substantially more accurate when compared with NIDSs. If the system-wide false positive rate is to be constant, this implies that the false positive rate for the devices, measured in false positives per unit of traffic, will also need to be two orders of magnitude better. (Note that since internal deployment of the same analyzers does not reflect using *independent* types of detectors, achieving this lower rate is not necessarily two orders of magnitude more difficult, as discussed below.) Additionally, because Hard-LAN devices will be composed with conventional perimeters, and may be composed with other Hard-LAN defenses, accuracy is critical and may require substantial further improvements.

Fortunately, because the traffic being monitored is on a LAN, there should be significantly less noise and accordingly fewer false positives when compared with an access link. Additionally, since the Hard-LAN approach is more tolerant of infections, this allows effective egress-attack monitoring, which is often a far more accurate signal with substantially fewer false positives. It is unclear whether the choice of observation points is sufficient for reducing the false positives to manageable levels, or whether different techniques will need to be developed.

VI. Towards Hard-LAN Designs

Although a substantial barrier, the stringent performance requirements for Hard-LANs are not insurmountable when designing anti-worm mechanisms. It should be possible for defenses to be either directly implemented or suitably modified for use in a Hard-LAN environment.

As an example of how targeting Hard-LAN environment affects algorithm and implementation design, we review the considerations we incorporated in designing a scan-detection algorithm specifically for scanning-worm containment (evaluated in greater depth in [10]). Scanning-worm containment operates by detecting a worm's scanning and then blocking the scanner, preventing a scanning worm from spreading throughout an institution.

Four our containment algorithm, we began with the Threshold Random Walk [6] scan-detection algorithm. The initial algorithm detects scans by observing the difference between the number of successful TCP connections and the number of failed connections. Three limitations prevented us from directly using this algorithm in a Hard-LAN environment: in its pure form, the algorithm is offline; it requires unbounded state; and it requires potentially unbounded memory access time.

The offline nature arises from the fact that the algorithm must distinguish between successful and failed connection attempts, yet to do so requires waiting for either some sort of server reply (a SYN ACK, accepting the connection, or a RST, refusing it) or a *timeout* indicating that no server host is present. This presents an immediate limitation, since the algorithm cannot, upon seeing an initial SYN, determine at that point how to behave. We avoid this limitation by changing the semantics of the difference between failures and successes. Rather than counting either a failure or success after the connection times out or completes, instead all new connections are immediately counted as failures, and only recast as successes when we observe a response. This adds a potential transient false positive which, fortunately, our trace-based assessment finds not to be significant.

Of greater concern for us is the memory required to track all pending and active connections. Since we wish to account for UDP traffic as well as TCP, then given the assumption that although UDP is connectionless, it is usually based on bidirectional behavior, we need to keep track of all active connections. If we desired perfect fidelity, this potentially requires unbounded access time and unbounded memory.

Thus, we approximate connection tracking using a fixed table with aliasing: an approximate cache which, rather than evicting collisions into a backing-store, simply combines entries. To look up a possible connection, we hash the connection's identifier to find the table entry, which contains both an age and status bits for whether the connection has been established in each direction.

Given the nature of the hash table, multiple connections may alias to the same value. However, it can be shown that this aliasing generally only creates false negatives, except for a very rare race condition. Accessing the table requires only a single read, and updating only requires writing to the same location, allowing approximate connection-tracking with only 2 accesses per packet, to only a single memory location. We have observed that a 1 MB table reaches only 10% capacity when monitoring the access link of a fairly large (several thousand hosts) site, allowing us to detect the 90% of the individual scanning attempts which do not alias to other entries in the table.

Likewise, we also developed a technique to track all external addresses of interest, using an associative cache. For this cache, we permute the address (using a 32 bit block cipher) and split the result to create both the index and the tag within the cache, resulting in substantial memory savings (we no longer need to record the IP address) and making the cache attacker-unpredictable. Again, this cache only needs a single read to perform a lookup, and a single write for an update.

An additional concern is how to manage evictions. We decided upon a policy of evicting the most-normal entry. Thus, addresses of particular interest won't be evicted. We observed that a 4 MB table is sufficient to track all external IP addresses of note, with only a few evictions, for the same site.

Using these tables, we were able to develop an efficient scan detection algorithm for both the inbound and outbound access links. We anticipate that this algorithm will be suitable for LANs as well. When a connection is established in only one direction, the corresponding address has a miss recorded against it. Establishment in the other direction changes it to

a hit. When the difference between the misses and the hits is greater than 10, the address is blocked because of its scanning behavior. We use the value 10 because it gives us an additional margin for error when compared with the value 5 in the origional TRW algorithm [6], which is based on perfect fidelity of the connection oracle instead of the "guilty until proven innocent" heuristic we developed.

The logic is sufficiently simple for a hardware implementation, as only 2 memory locations are updated per packet, and only requires a small (< 5 MB) total amount of memory. Thus, we believe our particular algorithm is feasible for integration within a switch's ASIC, or as a low-cost stand-alone network device. This is simply one example of how targeting the LAN's data rates and performance requirements affect the design of security algorithms. We expect that future algorithms will need different modifications when targeting this particular environment.

VII. Conclusions

Just as the nature of worms requires that resilient defenses are automatic, the nature of worms also constrains the location of these defenses. Since worms can easily cross coarse-grained perimeters like firewalls, the enterprise perimeter is an inadequate worm defense. Similarly, since we wish our defenses to continue to report after a machine is infected, end-host protection is problematic.

Thus we conclude that a large portion of the worm defenses will need to be embedded in the LAN, creating Hard-LANs designed to detect and respond to worms. The creating of Hard-LANs will require solving several technical challenges, in order to detect worms at very high data rates using low cost devices. Compared with conventional NIDS, Hard-LAN building blocks will require one to two orders of magnitude better cost/performance, and a similar improvement in the rate of false positives.

VIII. Acknowledgments

Funding has been provided in part by NSF under grant ITR/ANI-0205519 and by NSF/DHS under grant NRT-0335290. Dan Ellis was supported by the Active Worm Detection and Response MITRE-Sponsored Research project. Thanks to Michael Ellis and the reviewers for comments and suggestions.

References

[1] 3com. A new generation of embedded firewalls, http://www.3com.com/en_us/jump_page/embedded_firewall.html.

[2] CERT. CERT Advisory CA-2001-26 Nimda Worm, http://www.cert.org/advisories/ca-2001-26.html.

[3] CERT. Code Red II: Another Worm Exploting Buffer Overflow in IIS Indexing Service DLL, http://www.cert.org/incident_notes/in-2001-09.html.

[4] W. Cheswick and S. Bellovin. *Firewalls and Internet Security*. Addison-Wesley, 1994.

[5] M. Eichin and J. Rochlis. With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988. In *IEEE Computer Society Symposium on Security and Privacy*, 1989.

[6] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In *2004 IEEE Symposium on Security and Privacy, to appear*, 2004.

[7] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the Slammer Worm. *IEEE Magazine of Security and Privacy*, pages 33–39, July/August 2003 2003.

[8] D. Moore and C. Shannon. The Spread of the Witty Worm, http://www.caida.org/analysis/security/witty/.

[9] D. Moore, C. Shannon, G. M. Voelker, and S. Savage. Internet Quarantine: Requirements for Containing Self-Propagating Code. In *INFOCOM*, 2003.

[10] Nicholas Weaver and Stuart Staniford and Vern Paxson. Very fast containment of scanning worms. In *Scheduled to Appear, 13th USENIX Security Symposium*. USENIX, August 2004.

[11] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23–24):2435–2463, 1999.

[12] Snort.org. Snort, the Open Source Network Intrusion Detection System, http://www.snort.org/.

[13] S. Staniford. Networm.org faq, http://www.networm.org/faq/.

[14] S. Staniford. Containment of Scanning Worms in Enterprise Networks. *Journal of Computer Security, to appear*, 2004.

[15] Stuart Staniford and Vern Paxson and Nicholas Weaver. How to 0wn the Internet in Your Spare Time. In *Proceedings of the 11th USENIX Security Symposium*. USENIX, August 2002.

[16] Symantec. W32.blaster.worm, http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html.

[17] Trusted Computing Platform Alliance, http://www.trustedcomputing.org/home.

[18] VMware, Inc. VMware, http://www.vmware.com/.

[19] N. Weaver and D. Ellis. Reflections on witty: Analyzing the attacker. *;login:*, pages 34–37, June 2004.

[20] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham. A Taxonomy of Computer Worms. In *The First ACM Workshop on Rapid Malcode (WORM)*, 2003.

[21] M. M. Williamson. Throttling Viruses: Restricting Propagation to Defeat Mobile Malicious Code. In *ACSAC*, 2002.

CONFERENCE PROCEEDINGS

High-Performance Interconnects, Symposium on

Worms vs. perimeters: the case for hard-LANs

Year: 2004, Pages: 70-76
DOI Bookmark: 10.1109/CONECT.2004.1375206

Authors

N. Weaver, Comput. Syst. Lab., Stanford Univ., CA, USA
D. Ellis, Comput. Syst. Lab., Stanford Univ., CA, USA
S. Staniford, Comput. Syst. Lab., Stanford Univ., CA, USA
V. Paxson, Dept. of Comput. & Sci. & Eng., Ohio State Univ., Columbus, OH, USA

SHARE ARTICLE GENERATE CITATION

Abstract

Network worms - self-propagating network programs - represent a substantial threat to our network infrastructure. Due to the propagation speed of worms, reactive defenses need to be automatic. It is important to understand where and how these defenses need to fit in the network so that they cannot be easily evaded. As there are several mechanisms malcode authors can use to bypass existing perimeter-centric defenses, this position paper argues that substantial defenses need to be embedded in the local area network, thus creating "hard-LANs" designed to detect and respond to worm infections. When compared with conventional network intrusion detection systems (NIDSs), we believe that hard-LAN devices need to have two orders of magnitude better cost/performance, and at least two orders of magnitude better accuracy, resulting in substantial design challenges.

Trend Micro, Inc.
EX1011-8