Appears in HotOS-VIII (IEEE Workshop on Hot Topics in Operating Systems), May 2001.

Better Security via Smarter Devices

Gregory R. Ganger and David F. Nagle
Carnegie Mellon University

E-mail: {ganger,bassoon}@ece.cmu.edu

Abstract

This white paper promotes a new approach to network security in which each individual device erects its own security perimeter and defends its own critical resources (e.g., network link or storage media). Together with conventional border defenses, such self-securing devices *could provide a flexible infrastructure for dynamic prevention, detection, diagnosis, isolation, and repair of successful breaches in borders and device security perimeters. We overview the self-securing devices approach and the siege warfare analogy that inspired it. We also describe several examples of how different devices might be extended with embedded security functionality and outline some challenges of designing and managing self-securing devices.*

1. Overview

From all indications, assured OS security seems to be an impossible goal. Worse, conventional security architectures are brittle by design, because a small number of border protections (e.g., firewalls and/or host OSs) are used to protect a large number of resources and services. For example, an attacker who compromises a machine's OS gains complete control over all resources of that machine. Thus, such an intruder gains the ability to transmit anything onto the network, modify anything on the disk, and examine all input device signals (e.g., typing patterns and video feeds). Likewise, an attacker who circumvents firewall-based protection has free reign within the "protected" environment.

Having shared border protections for large sets of resources creates three fundamental difficulties: (1) the many interfaces and functionalities for the many resources (e.g., consider most multi-purpose OSs) make correct implementation and administration extremely difficult; the practical implications are daily security alerts for popular OSs (e.g., Windows NT and Linux) and network applications (e.g., e-mail and web); (2) the ability of successful attackers to freely manipulate everything beyond the border protection greatly complicates most phases of security management, including intrusion detection, isolation, diagnosis, and recovery; (3) having a central point of security checks creates performance, fault-tolerance, and flexibility limitations for large-scale environments.

This position paper promotes an alternative architecture in which individual system components erect their own security perimeters and protect their resources (e.g., network, storage, or video feed) from intruder tampering. This "self-securing devices" architecture distributes security functionality amongst *physically distinct* components, avoiding much of the fragility and unmanageability inherent in today's border-based security. Specifically, this architecture addresses the three fundamental difficulties by: (1) simplifying each security perimeter (e.g., consider NIC or disk interfaces), (2) reducing the power that an intruder gains from compromising just one of the perimeters, and (3) distributing security enforcement checks among the many components of the system.

Conventional application-executing CPUs will still run application programs, but they won't dictate which packets are transferred onto network wires and they won't dictate which disk blocks are overwritten. Instead, self-securing NICs will provide firewall and proxy server functionality for a given host, as well as throttling or labelling its outbound traffic when necessary. Likewise, self-securing storage devices will protect their data from compromised client systems, and self-securing graphics cards will display warning messages even when the window manager is compromised. In a system of self-securing devices, compromising the OS of an application-executing CPU won't give a malicious party complete control over all system resources — to gain complete power, an intruder must also compromise the disk's OS, the network card's OS, etc.

Augmenting current border protections with self-securing devices promises much greater flexibility for security administrators. By having each device erect an independent security perimeter, the network environment gains many outposts from which to act when under attack. Devices not only protect their own resources, but they can observe, log, and react to the actions of other nearby devices. Infiltration of one security perimeter will compromise only

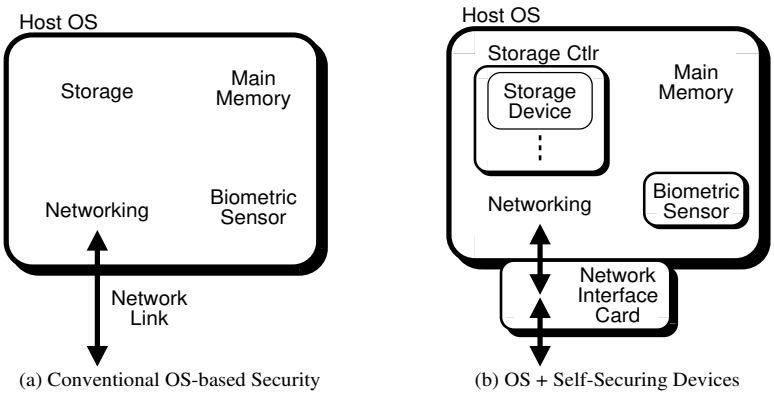


Figure 1. Two security approaches for a computer system. On the left, (a) shows the conventional approach, which is based on a single perimeter around the set of system resources. On the right, (b) shows our new approach, which augments the conventional security perimeter with perimeters around each self-securing device. These additional perimeters offer additional protection and flexibility for defense against attackers. Firewall-enforced network security fits a similar picture, with the new architecture providing numerous new security perimeters within each system on the internal network.

a small fraction of the environment, allowing other devices to dynamically identify the problem, alert still-secured devices about the compromised components, raise the security levels of the environment, and so forth.

Self-securing devices will require more computational resources in each device. However, with rapidly shrinking hardware costs, growing software development costs, and astronomical security costs, it makes no sense to not be throwing hardware at security problems. A main challenge for we OS folks is to figure out how to best partition (and replicate) functionality across self-securing components in order to enhance security and robustness. A corollary challenge is to re-marshall the distributed functionality to achieve acceptable levels of performance and manageability. After describing our inspiration for this architecture (medieval siege warfare), this position paper outlines some of our thoughts on these challenges.

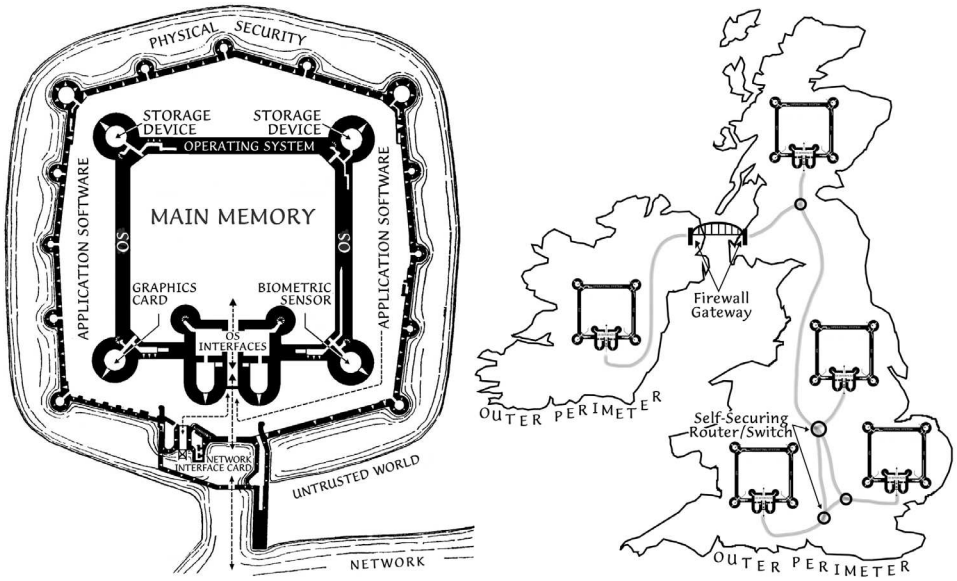
2. Siege Warfare in the Internet Age

Despite enormous effort and investment, it has proven nearly impossible to prevent computer security breaches. To protect our critical information infrastructures, we need defensive strategies that can survive determined and successful attacks, allowing security managers to dynamically detect, diagnose, and recover from breaches in security perimeters. Borrowing from lessons learned in pre-gun warfare, we propose a new network security architecture analogous to medieval defense constructs.

Current security mechanisms are based largely on singular border protections. This roughly corresponds to defense practices during Roman times, when defenders erected walls around their camps and homes to provide protective cover during attacks. Once inside the walls, however, attackers faced few obstacles to gaining access to all parts of the enclosed area. Likewise, a cracker who successfully compromises a firewall or OS has complete access to the resources protected by these border defenses—no additional obstacles are faced.[1] Of course, border defenses were a large improvement over open camps, but they proved difficult to maintain against determined attackers — border protections can be worn down over time and defenders of large encampments are often spread thin at the outer wall.

As the size and sophistication of attacking forces grew, so did the sophistication of defensive structures. The most impressive such structures, constructed to withstand determined sieges in medieval times, used multiple tiers of defenses. Further, tiers were not strictly hierarchical in nature — rather, some structures could be defended independently of others. This major advancement in defense capabilities provided defenders with significant flexibility in defense strategy, the ability to observe attacker activities, and the ability to force attackers to deal with multiple independent defensive forces.

[1] This is not quite correct in the case of a firewall protecting a set of hosts that each run a multi-program OS, such as Linux. Such an environment is more like a town of many houses surrounded by a guarded wall. Each house affords some protection beyond that provided by the guarded wall, but not as much in practice as might be hoped. In particular, most people in such an environment will simply open the door when they hear a knock, assuming that the wall keeps out attackers. Worse, in the computer environment, homogeneity among systems results in a single set of keys (attacks) that give access to any house in the town.



(a) a siege-ready computer system

(b) 2 siege-ready intranets

Figure 2. The self-securing device architecture illustrated via the siege warfare constructs that inspired it. On the left, (a) shows a siege-ready system with layered and independent tiers of defense enabled by device-embedded security perimeters. On the right, (b) shows two small intranets of such systems, separated by firewall-guarded entry points. Also note the self-securing routers/switches connecting the machines within each intranet.

Applying the same ideas to computer and network security, border protections (i.e., firewalls and host OSs) can be augmented with security perimeters erected at many points within the borders. Enabled by low-cost computation (e.g., embedded processors, ASICs), security functionality can be embedded in most device microcontrollers, yielding "better security via smarter devices." We refer to devices with embedded security functionality as *self-securing devices*.

Self-securing devices can significantly increase network security and manageability, enabling capabilities that are difficult or impossible to implement in current systems. For example, independent device-embedded security perimeters guarantee that a penetrated boundary does not compromise the entire system. Uncompromised components continue their security functions even when other system components are compromised. Further, when attackers penetrate one boundary and then attempt to penetrate another, uncompromised components can observe and react to the intruder's attack; from behind their intact security perimeters, they can send alerts to the security administrator, actively quarantine or immobilize the attacker, and wall-off or migrate critical data and resources. Pragmatically, each self-securing device's security perimeter is simpler because of specialization, which should make correct implementations more likely. Further, distributing security checks among many devices reduces their performance impact and allows more checks to be made.

By augmenting conventional border protections with self-securing devices, this new security architecture promises substantial increases in both network security and security manageability. As with medieval fortresses, well-defended systems conforming to this architecture could survive protracted sieges by organized attackers.

3. Device-embedded security examples

To make our new security architecture more concrete, this section gives several examples of how different devices might be extended with embedded security functionality. In each case, there are difficulties and research questions to be explored; here, we focus mainly on conveying the potential.

Network interface cards (NICs): The role of NICs in computer systems is to move packets between the system's components and the network. Thus, the natural security extension is to enforce security policies on packets forwarded in each direction [2]. Like a firewall, a self-securing NIC does this by examining packet headers and simply not forwarding unacceptable packets into or out of the computer

system. A self-securing NIC can also act as a machine-specific gateway proxy, achieving the corresponding protections without scalability or identification problems; by performing such functions at each system's NIC, one avoids the bottleneck imposed by current centralized approaches. NIC-based firewalls and proxies can also protect systems from insider attacks as well as Internet attacks, since only the one host system is inside the NIC's boundary. Further, self-securing NICs offer a powerful control to network administrators: the ability to throttle or tag network traffic at its sources. So, for example, a host whose security status is questionable could have its network access blocked or limited. Security administrators manage and configure self-securing NICs over the network, since they must obviously be connected directly to it — this allows an administrator to use the NIC to protect the network from its host system. By embedding this traffic management functionality inside the NIC, one enjoys its benefits even when the host OS or other machines inside the LAN border are compromised.

Storage devices: The role of storage devices in computer systems is to persistently store data. Thus, the natural security extension is to protect stored data from attackers, preventing undetectable tampering and permanent deletion [6]. A self-securing storage device does this by managing storage space from behind its security perimeter, keeping an audit log of all requests, and keeping previous versions of data modified by attackers. Since a storage device cannot distinguish compromised user accounts from legitimate users, the latter requires keeping all versions of all data. Finite capacities will limit how long such comprehensive versioning can be maintained, but 100% per year storage capacity growth will allow modern disks to keep several weeks of all versions. If intrusion detection mechanisms reveal an intrusion within this multi-week *detection window*, security administrators will have this valuable audit and version information for diagnosis and recovery. This information will simplify diagnosis, as well as detection, by not allowing system audit logs to be doctored, exploit tools to be deleted, or back doors to be hidden — the common steps taken by intruders to disguise their presence. This information will simplify recovery by allowing rapid restoration of pre-intrusion versions and incremental examination of intermediate versions for legitimate updates. By embedding this data protection functionality inside the storage device, one enjoys its benefits even when the network, user accounts, or host OSs are compromised.

Biometric sensors: The role of biometric sensors in computer systems is to provide input to biometric-enhanced authentication processes, which promise to distinguish between users based on measurements of their physical features. Thus, the natural security extension is to ensure the authenticity of the information provided to these processes. A self-securing sensor can do this by timestamping and digitally signing its sensor information. Such evidence of when and where readings were taken is needed because, unlike passwords, biometrics are not secrets [4]. For example, anyone can lift fingerprints from a laptop with the right tools or download facial images from a web page. Thus, the evidence is needed to prevent straightforward forgery and replay attacks. Powerful self-securing sensors may also be able to increase security and privacy by performing the identity verification step from within their security perimeter and only exposing the results (with the evidence). By embedding mechanisms for demonstrating authenticity and timeliness inside sensor devices, one can verify sensor information (even over a network) even when intruders gain the ability to offer their own "sensor" data.

Graphical displays: The role of graphical displays in computer systems is to visually present information to users. Thus, a natural security extension would be to ensure that critical information is displayed. A self-securing display could do this by allowing high-privilege entities to display data that cannot be overwritten or blocked by less-privileged entities. So, for example, a security administrator could display a warning message when there is a problem in the system (e.g., a suspected trojan horse or a new e-mail virus that must not be opened). By embedding this screen control inside the display device, one gains the ability to ensure information visibility even when an intruder gains control over the window manager.

Routers and switches: The role of routers and switches in a network environment is to forward packets from one link to an appropriate next link. Thus, one natural security extension for such devices is to provide firewall and proxy functionality; many current routers provide exactly this. Some routers/switches also enhance security by isolating separate virtual LANs (VLANs). More dynamic defensive actions could provide even more defensive flexibility and strength. For example, the ability to dynamically change VLAN configurations would give security administrators the ability to create protected command and control channels in times of crisis or to quarantine areas suspected of compromise. When under attack, self-securing routers/switches could also initiate transparent replication of data services, greatly reducing the impact of denial-of-service attacks. Further, essential data sites could be replicated on-the-fly to "safe locations" (e.g., write-once storage devices) or immediately isolated via VLANs to ensure security. Self-securing routers/switches can also take an active role in intrusion detection and tracking, by monitoring and mining network traffic. When an attack is suspected, alerts can be sent to administrators and to other self-securing devices to increase security protections. By embedding traffic monitoring and isolation functionality in self-securing routers/switches, one can enjoy its benefits even when firewalls and systems on the internal network are compromised.

Application-only CPUs: Though not strictly devices, most future host systems are likely to have multiple CPUs. They already have multiple functions, including OS-level resource management and various application-level tasks. Rather than trying to correctly implement and use a sandbox to safely host iffy code, we again suggest using physical boundaries — that is, run untrusted code on a separate *application-only CPU* that has no kernel (in the traditional sense) and no kernel-like capabilities. An application-only CPU should be physically locked away from its virtual memory mappings and device communication. The mappings, permissions, and external communication should be controlled by separate *management CPUs*, with which the application-only CPU communicates via a well-defined protocol. With such an organization, the safety of the hosted code becomes less critical, and the boundaries between it and more trusted components become more explicit.

4. Newly-enabled dynamic actions

Many new dynamic network security actions are enabled by the more numerous and heterogenous security perimeters inherent to the self-securing device architecture. To illustrate the potential, this section describes a few such actions:

Network DefCon Levels: Often, there is a trade-off between security and performance. For example, the more detailed and numerous the firewall rules, the greater the overhead introduced. Likewise, the more detailed the event logging, the greater the overhead. One use of the many new security perimeters is to support dynamic increases of security level based on network-wide status. For example, if an attack can be detected after only a small number of perimeters are compromised, the security levels at all other self-securing devices can be dynamically raised. As suggested above, this might take the form of more detailed firewalling at NICs, logging of network traffic to storage, and dynamic partitioning of the network into distinct VLANs.

Email Virus Stomping: One commonly observed security problem is the rapidly-disseminated e-mail virus. Even after detecting the existence of a new virus, it often takes a significant amount of time to root it out of systems. Ironically, the common approach to spreading the word about such a virus is via an e-mail message (e.g., "don't open unexpected e-mail that says 'here is the document you wanted'"). By the time a user reads this message, it is often too late. An alternative, enabled by self-securing NICs, is for the system administrator to immediately send a new rule to all NICs: check all in-bound and out-bound e-mail for the new virus's patterns. This would immediately stop further spread of the virus within the intranet, as well as quickly identifying many of the infected systems.

Traffic Throttling at the Source: As the previous example suggests, self-securing NICs allow network traffic to be throttled at its sources. Thus, a system that is deemed "bad" could have its network traffic slowed or cut off completely. Also, such malicious network activity as "SYN bombs" and IP address spoofing can be detected, terminated at its source, and even automatically repaired by the source's NIC (e.g., sending RST packets to clear SYN bomb connections).

Biometric Identity Verification: A more exotic use of self-securing devices is auxiliary identity checks on users. For example, imagine that an authenticated user does something that triggers an intrusion detection alarm. There are many possible explanations, one of which is that someone else is using the real user's session (e.g., while the real user is away at lunch). To check for this, a network security administrator could silently consult a nearby (or attached) self-securing video camera and perform face or iris recognition. Many other biometrics could also be used. The intrusion detection system could even trigger this check automatically and terminate the corresponding system's network and storage access, if the user is deemed to be an imposter.

Migration of Critical Data from Compromised Systems: If a system is compromised, one important action is trying to save and retain access to its user data. In our new architecture, this can be done by having the self-securing storage device (appropriately and authoritatively directed) encrypt and send the relevant contents over the network via the self-securing NIC. The self-securing router can forward the data to one or more alternate locations and route subsequent accesses to the data appropriately. In fact, different user bases could be routed to distinct replicas. With emerging device-to-device I/O interconnects, the storage-to-network transfer can be done with no host OS involvement at all, leaving the successful intruder with no way to stop it. Going back to the first example, another use of this support would be to frequently transfer the audit logs from various self-securing devices to on-line intrusion detection systems during perceived siege situations.

Displaying Trojan-defeating Messages: In perhaps the simplest example, a security administrator could direct a self-securing graphics card to override system directives and display a warning message. Such support would be particularly useful when users need to be warned to discontinue (or not start) using a system suspected of housing Trojan horses. Again, device-to-device communication allows this to happen over the network without host OS interference.

5. Research challenges

This change in network security architecture raises two major research questions, each with a number of sub-

questions. First, "what should each device do behind its security perimeter?" Answering this question will require exploration of cost, performance, and flexibility trade-offs, as well as exploring what is possible with the limited information available at any given device. Section 3 outlines potential functionalities for a number of devices. Second, "how does one effectively manage a large collection of independent security perimeters?" Answering this question will require exploration of tools and techniques for marshaling sets of self-securing devices, monitoring their current state, and dynamically updating their policies in the face of changes to and attacks upon the network environment's state.

The second question raises several complex subquestions that must be answered in order to realize dynamic and robust network security environments from large collections of distinct security perimeters. The clearest subquestions center on administrative control over the various devices, where security and convenience must be balanced. Research is also needed into how to reason about global network security given the set of local insights provided by distinct host systems and self-securing devices. Many other sub-questions exist, including those related to local policy configuration, robust reconfiguration, coordinated intrusion diagnosis, and avoidance of internally-imposed denial-of-service.

6. Related Work

Several researchers have used the siege warfare analogy to promote more comprehensive information security defenses [1, 3, 5]. Usually, the associated proposals are only loosely connected to the analogy, simply referring to the strengths (e.g., many parts), weaknesses (e.g., traitors), or eventual replacement of siege defenses. We use the analogy to inspire a specific defense strategy: use of physically-distinct barriers that monitor one another, defend collectively, and must be penetrated independently.

The concept of using physical separation of functionality for security is also not new. Perhaps the simplest examples are physically-secured machines with no network connections. Perhaps the best examples are firewalls and proxies, which enforce rules on network traffic entering and leaving an intranet via hardware specifically dedicated to this purpose. Here, we propose using physical component boundaries as the core of a security architecture rather than as a bandaid on inherently insecure network environments. The references below identify and discuss more related work.

7 Summary

This white paper promotes a new security architecture in which traditional boundary protections are coupled with security functionality embedded into self-securing devices. The resulting collection of independent security perimeters could provide a flexible infrastructure for dynamic prevention, detection, diagnosis, isolation, and repair of successful intrusions. Although many research challenges arise, we believe that the new architecture has great potential.

References

[1] Bill Cheswick. Security Lessons From All Over. Keynote Address, USENIX Security Symposium, 1998.

[2] David Friedman and David F. Nagle. *Building Scalable Firewalls with Intelligent Network Interface Cards*. CMU-CS-00-173. Technical Report, Carnegie Mellon Univeristy School of Computer Science, December 2000.

[3] Jr. John L. Woodward. Information Assurance Through Defense in Depth. U.S. Department of Defense brochure, February 2000.

[4] Andrew J. Klosterman and Gregory R. Ganger. *Secure Continuous Biometric-Enhanced Authentication*. CMU-CS-00-134. Technical Report, Carnegie Mellon Univeristy School of Computer Science, May 2000.

[5] Gary McGraw and Greg Morrisett. Attacking Malicious Code: A Report to the Infosec Research Council. *IEEE Software*, pages 33–41, September/October 2000.

[6] John D. Strunk, Garth R. Goodson, Michael L. Scheinholtz, Craig A. N. Soules, and Gregory R. Ganger. Self-securing storage: protecting data in compromised systems. *Symposium on Operating Systems Design and Implementation* (San Diego, CA, 23–25 October 2000), pages 165–180. USENIX Association, 2000.