Implementation of a Secure Web Environment for a Government Agency

Chad M. Steel

July 10, 2002

Abstract

This paper details the decision making process and implementation of a secure, multi-site redundant web hosting environment for a large government agency. The security objectives are detailed and the implementation of both the logical and actual security models examined. The system in question has extremely high visibility, and though it does not harbor classified information there would be a potentially severe economic impact to the country in the event of a security breach.[1]

Before

Overview

The government agency in question embarked on the deployment of a new web infrastructure to host their primary presence on the Internet. The legacy infrastructure suffered from a lack of scalability, contained multiple single points of failure, and did not have a reliable management infrastructure in place. From a security standpoint, the system was open to Denial of Service (DoS) attacks at each single point of failure, overall traffic flood attacks from a lack of appropriate bandwidth and hardware during peak load times, and vulnerability exploits from the lack of a consistent security update and response process.

Because of a poor overall architecture and after-the-fact security additions, there were significant drivers to upgrade the system. Given the sprawling, non-strategic organic growth of the existing system, there was no effective baseline to build upon. This led to a decision to redesign the system from scratch and avoid the pitfalls the original system fell into. Specifically:

- **Lack of Scalability.** There were non-burstable T3 connections in place to each of two datacenters in the legacy system. The maximum available bandwidth was limited by the incoming T3 circuits, with no quick upgrade path in the event of a bandwidth flood. Additionally, key pieces of the architecture had single servers performing critical path functions, with no provisions in the software or hardware for load balancing. This put the architecture at risk for attacks directed at single machines (DoS or other attacks.)
- **Multiple Single Points of Failure.** As mentioned above, there were multiple single points of failure in the legacy architecture. At the network level, there were single connections to the Internet terminating at a single router. The system was on a single switch with one network connection

from each device. The devices themselves, with a few exceptions, were not load balanced or clustered. Any failure at multiple layers through malicious or accidental means would result in a potential overall system failure.

- **Proprietary Security Infrastructure.** The use of bespoke code and in-house developed firewall and IDS technology in the legacy system allowed for protection against known vulnerability attacks on that infrastructure (since it was specific to this one system and not public, there were no known vulnerabilities!) Despite this advantage, the system relied upon security through obscurity. There was no independent verification or validation of the proprietary security infrastructure, and no way of knowing the true protection it afforded.

Based on the issues (security and otherwise) with the legacy infrastructure, a completely new architecture was designed from the ground up for the new system. The limitations of the existing infrastructure were evaluated and quantified, while at the same time new business requirements were gathered and taken into consideration when defining the system. Unlike the previous implementation, security considerations were taken into account starting with the business requirements and continuing through the logical and actual implementation models.

In terms of the general security goals of confidentiality, integrity, and availability, the latter two were of primary concern in this engagement. The system itself is open to the public, and disseminates critical information to a large audience. Because the information provided is a matter of public record and there are no individually identifiable information exchanges, confidentiality was not major driver (with the exception of backend administration.) Integrity was the primary concern – the falsification through defacement or replacement of the documents served by this agency would cause a large financial impact to the country as a whole. Additionally, the information provided needs to be available at key periods during the year without substantial delays, making availability a key concern.

As a baseline set of requirements for the system security assurance, National Information Assurance Certification and Accreditation Process (NIACAP)[2] guidelines were followed. The physical security components of the system were addressed separately through the previous certification of the data centers in which the systems are located and are not covered here. The NIACAP guidelines were used with an external audit as procedural guiding principles for final assurances on the system.

In addition to the NIACAP guidelines, ISO 17799[3] best practices were followed in security policy and procedure development. These practices were tailored in the form of custom created policies specific to this environment.

During

Logical Security Architecture

Given the high level objectives above, a logical security architecture was defined with a defense-in-depth strategy in mind. The logical security architecture was defined in co-ordination with the overall logical systems architecture, and the goals of the two taken into consideration when making all major component decisions. The key criteria for the logical security implementation were as follows:

- **Protection from defacement (Integrity).** All documents on the system should be unalterable by untrusted sources. File integrity should be maintained and examined on a regular basis. Content should never be altered (but it can be replaced from a trusted source) once it is published.
- **Protection from replacement (Integrity).** The number of trusted sources with the ability to publish content should be limited to the fewest possible. The publishing mechanism should be separate from the delivery mechanism.
- **Protection from Denial of Service (Availability.)** The system should contain no single points for failure to reduce risk of accidental or malicious Denial of Service (DoS). Appropriate filtering should be in place to prevent spoofing and other DoS techniques. Incident response mechanisms should specifically address DoS attacks.
- **Protection from Distributed Denial of Service (Availability.)** IP spoofing should be prevented. Traffic patterns indicative of Distributed Denial of Service (DDoS) attacks should be captured early. Available capacity for all system components (HTTP, DNS, FTP, etc…) should be high enough to reduce the immediate impact of a DDoS attack, even at peak loads.
- **Early warning of potential threats (Integrity, Availability.)** Intrusion detection mechanisms should be in place to quickly detect serious probing or threat activities. Intrusion response plans and monitoring of these threats should be documented prior to occurrence.
- **Restriction of Access (Integrity.)** Access should follow a least privilege model. All public access should be restricted to FTP, HTTP, SSL, and this access enforced at multiple layers. Administrative access should be limited to individual machines and root access held by only two individuals (primary and backup.) Administrative network access should be through a separate infrastructure
- **Adverse event auditing/logging (Confidentiality, Integrity.)** Any adverse activity detected should be logged and stored for later forensics. Logs should have restricted access, be robust enough to support investigation and prosecution, and be unalterable after initial creation. Regular, secure offisite storage of logs for multiple years after generation

should be accommodated. Auditing policy should balance system performance with security concerns.

- **Separation of administration (Confidentiality, Integrity, Availability.)** All administrative functions should be performed through a separate administration network, including backup, system monitoring and reporting, and updating. Alerting should be available, as well as administrative access to the primary severs, even when there are network issues with the primary access network.

Based on the above criteria, the following logical architecture design was created. The logical design was translated into an actual physical design during the implementation. Figure 1 shows the logical component parts of the security and their relative protection levels against both sophisticated and unsophisticated attacks.

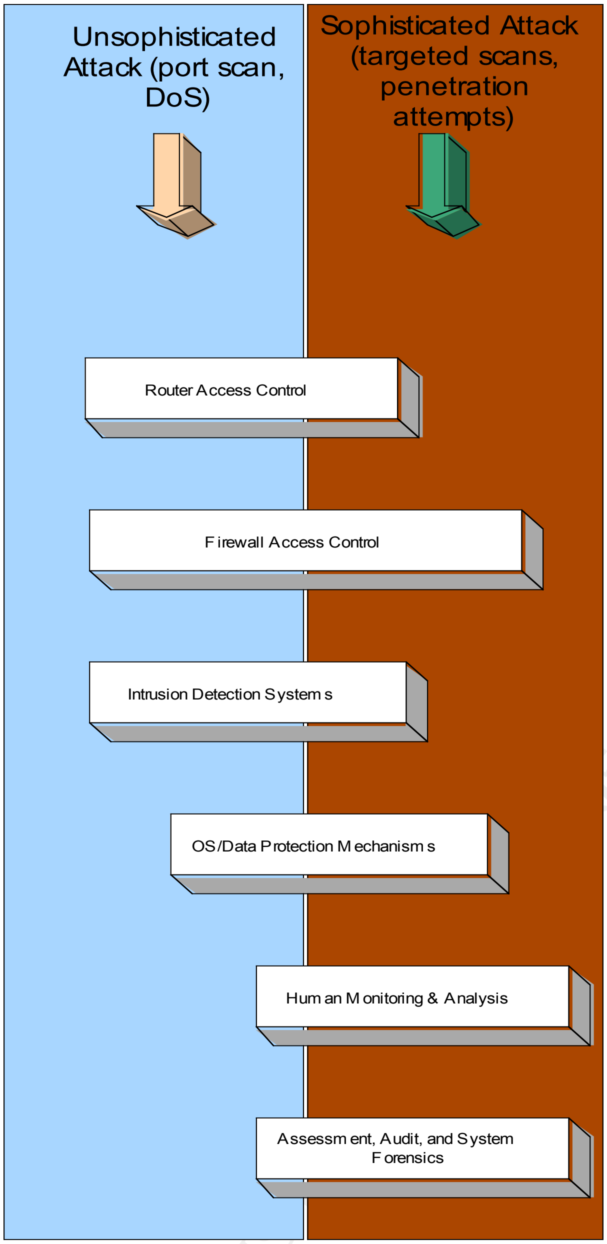
Unsophisticated Attack (port scan, DoS)
Sophisticated Attack (targeted scans, penetration attempts)
Router Access Control
Firewall Access Control
Intrusion Detection Systems
OS/Data Protection Mechanisms
Human Monitoring & Analysis
Assessment, Audit, and System Forensics

Figure 1 - Logical Protection Mechanisms

The logical protection mechanisms shown in Figure 1 were put in place to address the security design goals stated above. The logical protection mechanisms were defined as follows:

- **Router Access Control.** Any broad anti-spoofing rules (standard reserved addresses) are implemented here at the upstream router - specific IP's are blocked at the firewall level. Additionally, the router itself has capacity expandable to an OC192 on its incoming trunk line, allowing for protection against bandwidth flood attacks.
- **Firewall Access Control.** Granular IP address blocking (specific internal IP addresses, individual IP's of known scanning machines, class C's of

violating networks) is performed by the firewall layer. Ports 21, 80, and 443 are the only inbound ports allowed, and 25 is the only outbound port permitted to the Internet.

- **Intrusion Detection Systems.** Both network-based and host-based intrusion detection capabilities are covered under this logical component. Network-based intrusion detection uses a signature-based, heuristic, and statistical approach to monitoring network traffic. Host intrusion detection focuses on changes to file structure and auditing/alert generation based on anomalous events.
- **OS/Data Protection Mechanisms.** Data itself is protected from modification by OS and disk subsystem controls. OS hardening and minimization of the OS footprint is performed on all systems, as well as application hardening (specifically on the web servers.)
- **Human Monitoring and Analysis.** A specific security manager is assigned solely to oversee security on this implementation. This individual co-ordinates responses from two separate monitoring teams (24x7 host and network based teams – both of which perform human analysis on all alerts) and is the single point of accountability for all security issues.
- **Assessment, Audit, and System Forensics.** All important activities are audited to a secure, separate logging server. These are used with pre-defined processes for system forensics as necessary. A pro-active assessment process is in place with full, quarterly vulnerability assessments of the entire system infrastructure by an independent security team.

The logical protection mechanisms defined above were translated into actual protection mechanisms, products, processes, and procedures in the implementation. These choices were made along with the systems architecture team to meet all of the implementation needs – security included.

Architecture Implementation

The actual security architecture was defined based on the logical areas set forth, the network constraints of the infrastructure and associated applications, and the business requirements identified in the analysis phases. Product selection was performed for each key component, and network design choices made based on differing security zones. The key security product choices were for firewalls and network-based IDS as follows:

Firewalls – The firewalls selected were Checkpoint Firewall 1 stateful inspection boxes running on the Nokia hardware and IPSO OS[4]. Checkpoint allowed for centralized management of all devices at two locations with consistent policy enforcement from a common repository. Due to the high throughput of this application, proxying solutions were not viable. Because the major network components were Cisco-based, we selected Checkpoint over Pix for diversity in protection products.

IDS – A modified Network Flight Recorder (NFR)[5] implementation was chosen as the network-based intrusion detection platform. The IDS sensors were implemented to have heuristic, signature-based, and statistical checks for intrusions, and were connected in non-addressable (no IP address), promiscuous mode on key network segments. The backend interface is connected through a management LAN to a 24x7 human/machine alerting and response mechanism.

The overall physical architecture of the site is shown in Figure 2 below. All networks were segmented into Virtual LAN's (VLAN's) based on their relative security zones. The defined VLAN's are as follows:

VLAN 1 – This is the primary VLAN (and linked to VLAN 2 for monitoring via port spanning) for monitoring non-load balanced traffic. All raw requests coming through the firewalls are monitored by the IDS before being passed through the load balancing devices.

VLAN 2 - This is an extension of VLAN 1 for the secondary data feed. VLAN 2 is additionally able to operate on it's own in the event of a component failure with VLAN 1 and assists in global load balancing (not depicted.)

VLAN 3 – The third VLAN is the primary network segment for serving up information. Web, search, and FTP servers all have similar levels of security from the standpoint of access from the Internet, and have been grouped together into one security zone. The second IDS is placed on VLAN 3 to monitor both internal traffic (for compromised boxes or insider attacks) as well as Internet traffic. Because the IDS on VLAN 1 is in front of the load balancers, it cannot resolve the load balanced IP address to a specific machine. IDS 2 allows us to determine which specific IP was hit with an attack for quicker response and forensics.

VLAN 4 – A secondary storage VLAN was configured for access to the mass storage device (EMC). The VLAN was setup exclusively as a conduit for reading from/writing to the shared storage from production. All access to the mass storage is firewalled by an additional pair of load balanced firewalls.

VLAN 5 – Database and application servers were segmented on the same VLAN. The databases are currently utilized as read-only from the web segment, and read/write from the application servers. The application servers are proxied through the web servers, with requests passing through the backend firewalls on a separate port (not 80 or 443.)

VLAN 6 – The backend content management and publication systems were placed in a final VLAN (in actuality a separate LAN with physically separate switches.) This VLAN has read/write access to the mass storage device, which acts as a gap technology to prevent direct access to backend systems from the web (this would require compromising a web server, then a firewall, then the EMC unit itself to happen.)

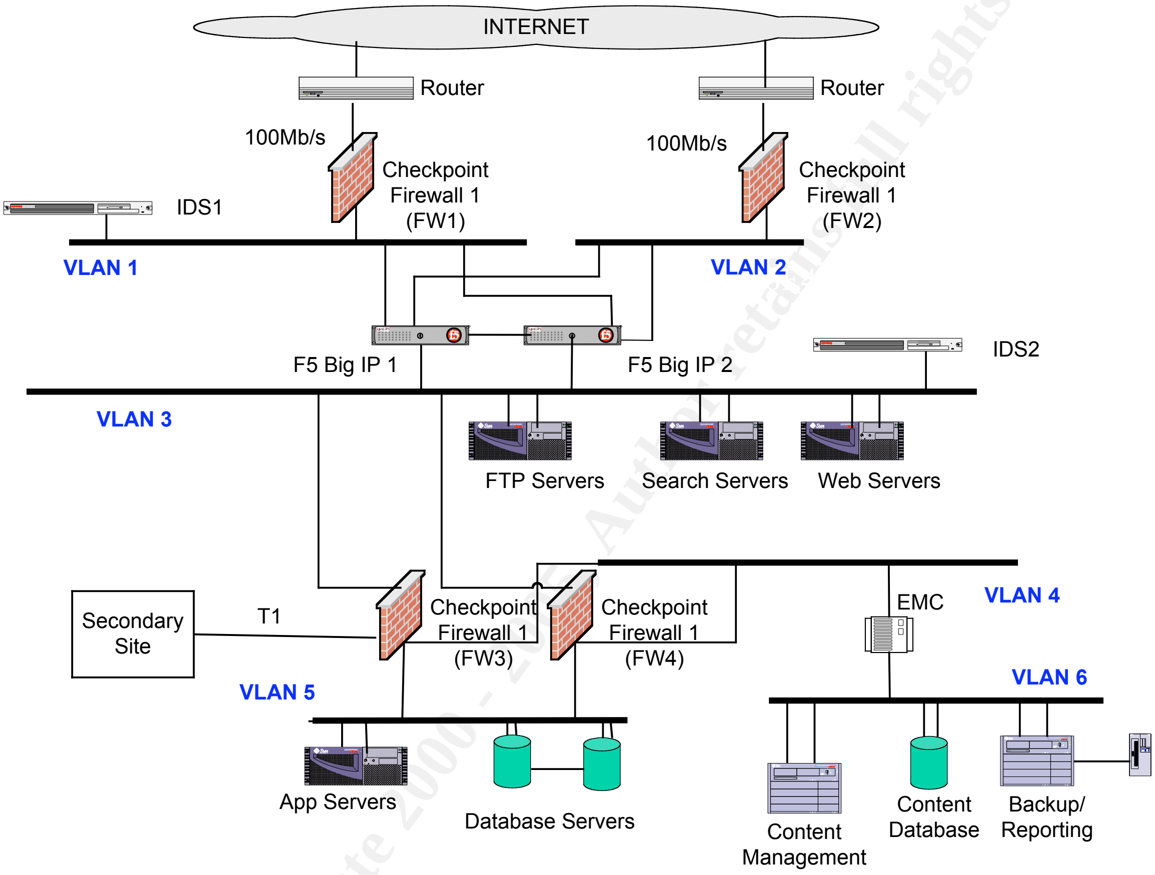

Figure 2 - Physical Architecture

Implementation Features

There were several “features” implemented which enhance the overall security of the infrastructure. These are not individual security components, but rather techniques and configurations that assist in creating a secure environment. Techniques include:

Secure content publishing. All content management and publishing is performed on a physically separate content network. The content management system pushes out the information once it is ready for publication to a read/write partition on the EMC for testing. After testing is complete, a replication from the read/write partition to a read only partition is performed. The web servers and FTP servers have this shared partition as their root – not only is it read only, but on a separate platform. Because of efficient caching, we are able to use the shared storage to serve end user content with no performance degradation.

Extended verification. We utilize the extended verification option of the load balancers to validate page response times. Within the extended verification, we are able to identify specific page components on a GET request from the HTTP servers. If any server suffers a major defacement, the page components will be altered or not present and the server will be taken out of the rotation automatically by the load balancers.

Network Address Translation (NAT). Internally we use NAT with different address bands for the differing component layers. The externally visible IP addresses are translated at the load balancer layer to reduce the amount of information provided by the site to the public.

System Minimization. For all of the systems in this implementation, we minimized the footprint to include only the necessary services. All additional local and network services were turned off, and local firewalling was enabled with a tool similar to IPChains for further lockdown. Finally, all unnecessary accounts were disabled, and existing accounts provided only the minimal privileges necessary for operation.

Functional Separation. All of the key servers on the client-facing systems were segregated according to security level and operational function. This allowed us to minimize the impact of a vulnerability on a given service (e.g. an FTP vulnerability that was exploited would not be directly impact the HTTP servers) and localize issues to a given segment.

Post-implementation, we performed a detailed vulnerability assessment of the infrastructure. Our own assessment included automated scans from three separate tools, a manual penetration test of the environment, and architecture reviews with a separate security team. After the assessments, all findings were immediately remediated and reassessed for compliance. [6]

In addition to the internal assessment, two additional assessments were performed by external entities. The recommendations from these assessments were incorporated into future enhancement plans for the environment.

As security is a moving target, we continue to perform assessments on a quarterly basis to uncover new vulnerabilities or bugs since the previous attempts. The systems administrators are held responsible for ensuring patches are applied in a timely manner and are held accountable in their bonus numbers for timely patch installation.

Incident Response

All incidents or alerts that are generated are monitored by a computer and human reporting system. The computer system tracks all alerts as they occur, and classifies them initially as high, medium, or low priority. Alerts such as an automated scan of the FTP port would generate a low priority, whereas a Denial of Service signature would generate a high priority alert.

All of the alerts are reviewed by a 24x7 security monitoring team. If the alerts are found to be valid and represent a threat to the environment (even a minor one), they are classified as incidents and the incident response procedures are put in place.

The incident response procedures are pre-written for various scenarios (Denial of Service, Defacement, etc…) and are followed under the direction of the site's Sr. Security Manager. The Sr. Security Manager classifies the incident, directs the investigation, and makes the appropriate notifications/takes the appropriate actions. The entire process is documented and stored for later evidentiary and reference usage.

After

Assessment Results

Post-implementation, the infrastructure was validated and verified with a series of tests and audits (in addition to the functional component testing during the implementation.) Testing on the completed system from a security standpoint involved automated assessments and manual confirmation of settings against checklists. Additionally, performance testing was done to ensure scalability to the levels expected by projections to protect against bandwidth flooding and DDoS attacks.

The vulnerability assessment was performed by an independent internal security team. The testing was performed at three specific points – outside the firewall, from the perspective of a hacker, inside the firewall, from the standpoint of an insider, and on the management network, simulating an attack from an administrator.

All testing was done with three different vulnerability assessment tools. In our testing, no one tool consistently identified all vulnerabilities in a given system, and the use of multiple scanning tools increased the coverage and substantially reduced the risk of missed holes. The tools chosen for this engagement were Nessus, SSS, and ISS Internet Scanner.

Nessus is the leading open source security scanning tool. Covering a large number of vulnerabilities on a substantial mix of systems, Nessus was ideal for this project. Additionally, Nessus is publicly available at no charge, making it the tool of choice for both hackers and auditors.

SSS, or Shadow Security Scanner, is an up-and-comer in the vulnerability assessment tool market. Available as shareware for a reasonable charge, it is readily accessible to small companies or funded hackers. The coverage of SSS is actually claimed to be greater than either Nessus or ISS in the number of vulnerabilities checked. In our testing, we've found SSS to find holes not found by the other tools, but also to return larger numbers of false positives (mis-identified, valid functions which are thought to be vulnerabilities.) The largest number of false positives occur when improper error codes are returned from a server (e.g. a 404 error page with a 200 return code.)

ISS is the leading commercial security tool available and holds the largest market share for commercial scanners. Along with System Scanner and Database Scanner (which we also used internally), ISS Internet Scanner provides a comprehensive look at the overall security of a system from the OS through web application layers. ISS is expensive to license, and therefore only readily available to large companies and government agencies (though older version have made their way into the "wild.")

The results of all three tools on the sites were as expected. There were no vulnerabilities found from the Internet side, and the internal vulnerabilities identified were minimal and immediately patched (these were primarily related to the original implementation of NFS and an older version of SSH which were originally present.)

After the tools, manual checklists were followed on each of the machines to confirm configuration settings. The checklists were based on the hardening guidelines implemented by the administration team from SANS, NIST, and the NSA (in addition to several proprietary hardening guidelines used internally.) The independent review team used the same audit checklists as the install team to confirm the configurations.

In the manual audit, only one misconfiguration was uncovered and fixed. The web servers allowed for directory listings on proxied requests – because the proxied requests were being delivered by the application server, the setting to disallow directory listings was never applied to these boxes (it was applied only to the web and search servers initially.) This vulnerability was minor, but an example of the type of issue that is not found by automated scanning tools.

After initial security testing, performance testing was done using Mercury Interactive's LoadTest software.[7] This testing was carried through to over 100,000,000 page views a day, with the limiting factor being the incoming bandwidth.

With the knowledge of the likely failure point in a DDoS attack being the incoming bandwidth, we were able to perform three post-implementation actions in response. The first was obtaining the necessary permissions to use the upstream routers to block IP's through the use of ACL's as needed to prevent the bandwidth from being used behind them. The second action was the implementation of a DDoS response mechanism in the Incident Response Procedures to deal specifically with bandwidth flood attacks. Finally, a plan was put in place to upgrade quickly from 100Mb/s incoming connections to 1Gb/s incoming connections as necessary.

After security testing, the systems were given the green light to go live. Post-production operations were monitored, and lessons learned gathered from the entire security implementation lifecycle were documented.

Post-Production Operations and Lessons Learned

The above-mentioned system has been live and in production for a period of six months at the time of writing. There have been numerous external audits by both government entities and third parties, and all have been successfully passed. Additionally, although there have been extensive penetration attempts (both authorized and unauthorized), there have been no successful intrusions. Based on the experiences in developing the architecture and the post-launch period of management, there are several best practices/lessons learned that are worth highlighting:

- **Design Security from the Start.** It's been said many times before but bears repeating – get security involved from project inception. Several of the decisions on networking and systems architecture were driven primarily from a security standpoint. If security hadn't been taken into consideration from the initial phases, many of the features implemented would have been costly and infeasible to implement after the fact.
- **Monitor Internal and External Addresses with IDS.**[8] Both internal and external addresses should be resolvable by at least one IDS system. An external IDS alone cannot resolve load balanced/hidden IP addresses for specific systems. In the event an exploit is attempted, all servers in the load balancing scheme must be examined if there is no adequate mapping of attacks to the physical servers. Additionally, any intra-site or internal attacks will not be captured by an external IDS alone.
- **Prepare Response Plans Ahead of Time.** All security administrators know the form several likely attack scenarios are going to take – virus infection, DoS, DDoS, page defacement, etc… For each of these scenarios, it makes sense to prepare a response ahead of time. Included should be detailed action plans, as well as key contact information (when a DoS is actively occurring is NOT the time to go looking for the network administrator's pager number!)
- **Run Scanning Tools Yourself.** Nessus[9], one of the best vulnerability assessment tools available, is free. Shadow Security Scanner (SSS), another excellent tool, is available at a minimal cost. There is no reason

not to run these tools on all development efforts, both internally and externally, prior to launch and fix the holes they find.

- **Document all Procedures.** All operating procedures related to security, down to when, how often, and who performs the actions needs to be documented. This is especially true for sites that need to pass a government audit – if you don't have a documented procedure for a security task, it is assumed that you don't do it![10]
- **Make Security Part of All Bonus Objectives.** The best way to get systems, network, and application personnel to take notice of security is to make their bonuses contingent upon it! A quarterly security assessment can coincide with bonus decisions – once a bonus is lowered once for not patching a system or forgetting to turn off a service, the message will be understood loud and clear!
- **Separate the Management Network from the Production Network.** Dual home production systems and provide a separate network with different hardware and its own addressing scheme for the management tasks. Not only will this minimize the performance impact of activities like logging, but provides you an alternate channel for alerting and connecting when a DoS attack is launched.

The security implementation for this endeavor is not completed even post-launch. Security is always a moving target, and ongoing efforts outside of the day-to-day security management and response include architecture re-reviews, product evaluations, and procedure updates based on industry changes. Additionally, future application changes are implemented with higher security from the start by training application developers in techniques such as those documented by the Open Web Application Security Project (OWASP)[11], ensuring continued high levels of infrastructure security.

EndNotes

[1] Steel
[2] Hayden
[3] NIST, p. 1-2
[4] Nokia
[5] Frederick
[6] Herzog
[7] Steel
[8] Frederick
[9] Renaud
[10] Hayden
[11] OWASP

References

Frederick, K. “Network Monitoring for Intrusion Detection” SecurityFocus. August, 2001. URL: http://online.securityfocus.com/infocus/1220 (July 9, 2002)

Hayden, M. “National Information Assurance Certification and Accreditation Process” NSA. April, 2000. URL: http://www.nstissc.gov/Assets/pdf/nstissi_1000.pdf (July 10, 2002)

Herzog, P. et al. “The Open Source Security Testing Manual v1.5” IdeaHamster. May, 2001. URL: http://www.ideahamster.org/osstmm.htm (July 6, 2002)

NIST. “International Standard ISO/IEC 17799:2000” NIST. December, 2001. URL: http://csrc.nist.gov/publications/secpubs/otherpubs/reviso-faq.pdf. (July 7, 2002)

Nokia. “Firewall Solutions, Nokia and Checkpoint” Nokia Corporation. July, 2002. URL: http://www.nokia.com/securitysolutions/network/firewall.html (July 10, 2002)

OWASP et al. “A Guide to Building Secure Web Applications and Web Services” OWASP. June, 2002. URL: http://www.cgisecurity.com/owasp/OWASPBuildingSecureWebApplicationsAndWebServices-V1.0.pdf (July 9, 2002)

Renaud. “Reduce the Costs of an Audit with Nessus 1.2” Nessus. July, 2001. URL: http://www.nessus.org/pres/bh2001/index.html (July 8, 2002)

Steel, C. “Case Study of a Multi-Site Redundant Web Hosting Environment” IEEE Internet Computing. September, 2002.