Global Information Assurance Certification Paper

Case Study
Implementing a Process to Validate And Respond to Security Issues

Abstract

As a software vendor of an integrated system whose products were developed when speed, convenience and efficiency were our customers' primary concerns, we find ourselves under significant scrutiny to address security issues. We receive requests in a variety of forms from simple emails to extensive scan reports. Our team was tasked with four major goals: process, verify, respond, and document security issues. By developing and documenting a Security Queue, running vulnerability scans utilizing Internet Security Systems (ISS) Scanner and Nessus, and performing UNIX hardening activities, the paper will show how we met the goals and measured our progress.

Winston Burgess
Date submitted 06/01/04
GSEC Version 1.4b Option 2

The Problem

An extensive series of initiatives has been started at our company to respond to the growing security concerns of our customers. These initiatives span every department, Marketing, Development, Sales, Support, Information Technologies, Human Resources, Legal and others. As part of a Global Technical Services (GTS) organization, I was required to respond to customer queries forwarded onto us by both our presales and post sales organizations in regards to security of our UNIX based products. GTS was faced with the challenge of prioritizing and classifying requests, and providing responses that ranged from a single line email to a full product customization and deployment engagement.

Our products span multiple architectures but share similar characteristics:

- Client/Sever system architecture
- Have databases that could potentially store sensitive data
- Must respond to "real-time" events
- Require significant interaction with other customer systems, networks and databases.

Various versions of our software span multiple revisions of the Solaris OS. Adding to the complexity of the integration is a real-time environment implemented through distributed servers, which puts significant constraints on what can and cannot be changed in the configurations of the systems.

The Growing Storm

For a couple of years I had been receiving queries from our customers passed on from Sales, Deployment and Support Teams in regards to security issues with our products. First they trickled in one per quarter, then monthly. Finally a week didn't go by without multiple requests to review scan reports, apply an OS patch, turn of a service, or to add some security enhancing application.

Prior to the IT focus on security, our systems were perceived as a black box by some of our customers and their IT departments. The customer IT departments had begun to scrutinize the systems, performing vulnerability scans, and installing various security packages on the systems. It was clear that we needed a different approach and a way to address our customers' security concerns.

In many cases these requests were being submitted by the customer's business unit, which had no idea what they were asking. They were only responding to a security initiative forced upon them by their IT or corporate security office. They would pass those requests on to us to satisfy their requirements for compliance. We needed to be able to not only respond but also educate our customers and justify our responses.

Initially as the first security requests came in, we embarked on a series of expensive but effective customizations based on specific customer requirements for securing our system. It was clear that for each customer there were different approaches to how they wanted to secure our systems and what they were willing and not willing to do. There was no single response that we could produce that could be generically rolled out to everyone else.

The Security Team in the Global Technical Services (GTS) organization was tasked to assist both pre-sales and post-sales organizations in our company.

Specifically our goals were as follows:

- Develop and document a process to respond to Sales and Support questions on security issues
- Verify customer requests such as scan reports or specific vulnerabilities
- Respond to specific hardening/customization engagements or requests
- Document security related issues and processes

While we all knew that this "security thing" was out there, we did not know the extent and degree to which it would affect us. Initially it was visible through a small number of large customers whose IT departments had begun to focus attention on our systems that were on their networks. As these requests increased and diversified, it was clear that the problem needed to be addressed at a variety of levels.

The Approach

We took on several initiatives to allow our company to address the growing focus on security issues from a support perspective.

- Development of a Security Queue to respond to requests on security issues
- Verification of security vulnerabilities in our systems
- Development and testing of hardening procedures on our systems
- Documentation of vulnerabilities and resolutions

These initiatives would contribute to the overall security policies and direction of our company and its products while addressing the immediate security concerns of our customers.

Security Queue

The Security Queue allowed internal resources to submit requests to Subject Matter Experts (SME) based on the platform or solution that the question was about. My first challenge was to make the queue effective and understood by documenting it. Diagram 1 represents the flow of that queue.

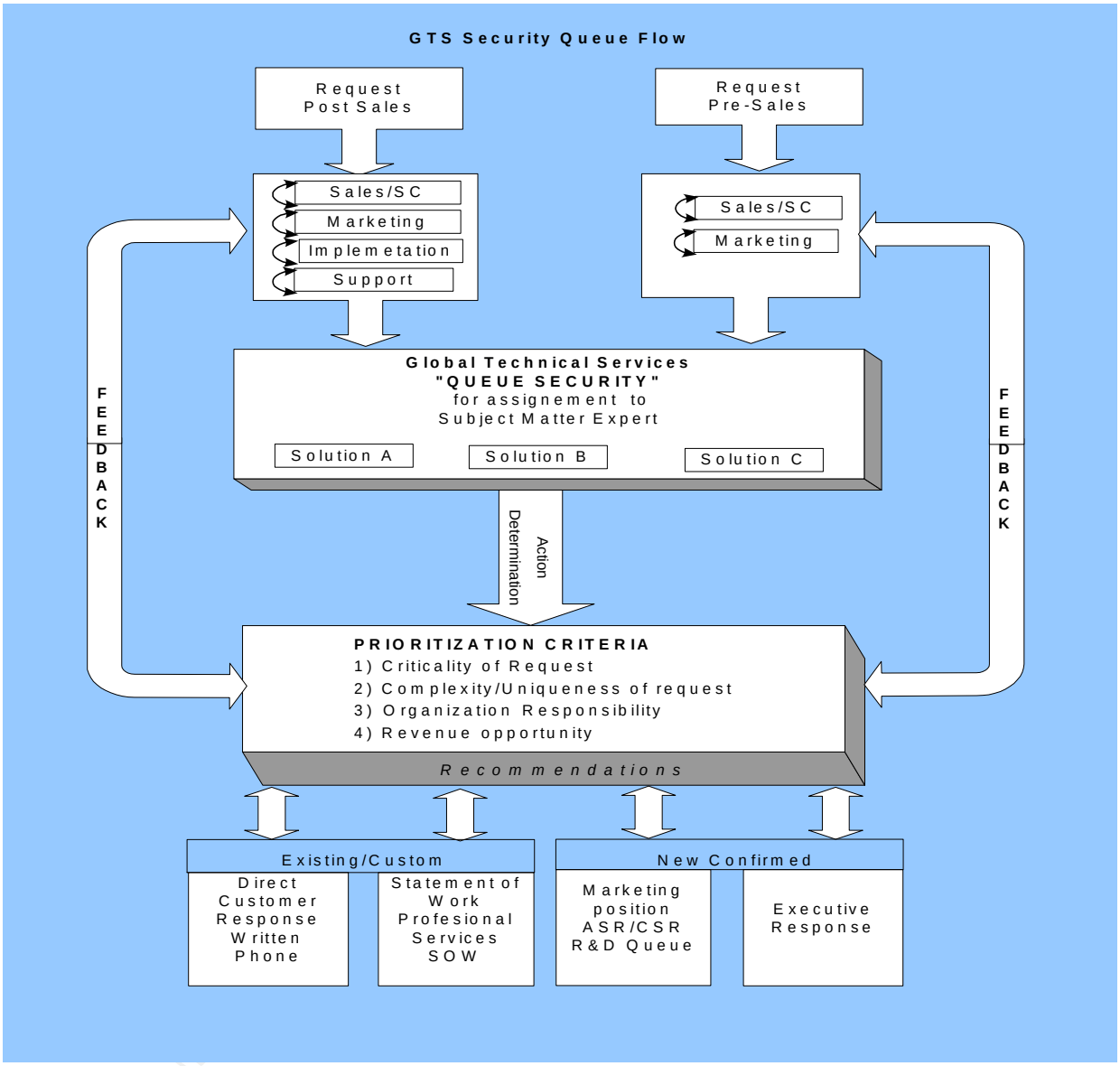


Diagram 1 GTS Security Queue Process Flow

Requests could come in from both a pre-sales and post sales environment. A representative of our company entered the request to the Security Queue. The Security Queue Process Flow encouraged internal organizations to first review the request amongst themselves to determine if a request was a valid security question and to see if resources within their group may be able to respond to the query. This is represented by the small curved arrows within the requesting groups.

The requesting group was required to fill out a submission forms via MS Outlook predefined form[1] or via a web page which would generate an MS Outlook request. This

process standardized requests and insured that sufficient information was attached that allowed for classification and assignment to the SME.

Figure 2 is an example of the form as it would be received by the Security Queue team members. The required information is self explanatory. Requiring the form helped to minimize one line requests like “please help me with my customer’s security problem.” (Yes, that is an actual email I got!) One of the fields, “Primus Search” referred to the company knowledge base. We wanted internal resources to search there first for an answer. Further details on the roll of the knowledge base in responding to security questions will be detailed in the Documentation Section below. Once submitted, the query was reviewed for completeness and accuracy and assigned to the appropriate Subject Matter Expert (SME) for a response by the Security Queue Team leader.

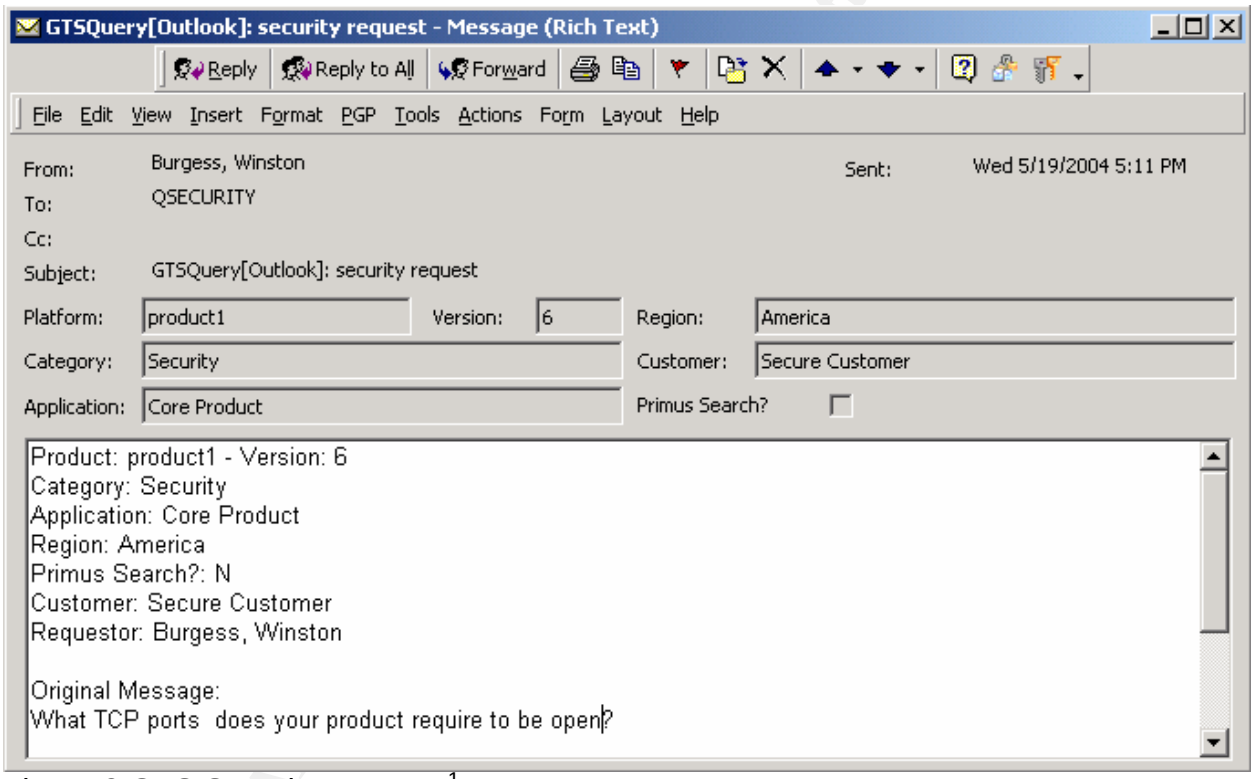


Figure 2 GTS Security Request[1]

The SME reviewed the request and determine an initial response based on the provided information. Queries that were well documented and precise in the request received faster more accurate responses. Queries that were vague often required follow-up and an exchange of information between the SME and the requesting party before an appropriate response could be delivered. This is represented by the *Feedback* Arrows in the Process Flow diagram.

Each SME had latitude on how they would prioritize and respond to requests. As the SME in charge of the UNIX based products, I chose four prioritization criteria as shown in Security Queue Process Flow above:

1) **Criticality of request** High, Medium, Low, as determined by GTS management and myself. I used a Qualitative Assessment[2] to asses the risk
    a. A system had a severe vulnerability.
    b. The threat was great enough to have a high probability of being exploited.
    c. The threat could compromise either our customer's or our company's interest.
2) **Complexity/Uniqueness** Requests that are simple or that have been addressed previously can be responded to quickly. Requests that are more complex to resolve, unique, or involve significant research require more time and resources to resolve.
3) **Organizational responsibility**. In some cases requests are better responded to by a different organization. For example, queries about our corporate security polices may be referred to our Policy Group or questions on product futures in regards to security may be referred to Marketing for a response.
4) **Revenue opportunity**. As a business, all requests are evaluated for revenue opportunity. Revenue may be direct or indirect as a result of the activity necessary to respond to the request.

Once a request has been prioritized, the SME researched and determined the best way to respond. The SME becomes the focal point coordinating activities necessary for a response. Some cases are simple and involve no more than a phone call with the customer to clarify an issue. In other cases a written response is required.

Response types were split into 2 categories

- Existing and custom requests
- New confirmed requests

The primary reason for these 2 categories was ownership and escalation. Global Technical Services and the SME could handle and complete responses for the first type, those issues that are known to exist or are requests for customization. The SME is responsible for reviewing new issues, confirming them as truly new and requiring escalation. However, a new critical vulnerability not yet known or a significant product security issue (for example a CERT advisory) would require any response to be approved by an Executive Review Board.

As I reviewed the questions and the vulnerability scan reports coming into the queue, I realized I needed to verify that the vulnerabilities being identified truly existed and gain an understanding of how the systems were being examined and the reports generated. This would allow me to build a library of responses as well as confirm the existence of the vulnerabilities.

Verification

I began to research on the web for scanning tools. I wanted to evaluate the tools and find out as much about how they worked as I could.

Internet Security Systems (ISS) Internet Scanner

I found a great tutorial on ISS Internet scanner at the Georgia Tech website[3]
This provided an excellent overview of the ISS Internet Scanner tool, what it did, and how to initially set it up and run a system scan.

Having read the introduction, I went to the ISS website[4.] Many of the requests submitted contained an ISS Scanner report. The query usually went something like, "Please respond to the attached report," and then there was an 80 page report attached.

I found at the ISS site a 30 day restricted evaluation license of the Internet Scanner product. I loaded the software on to my Windows 2000 PC. The software was easy to load and install with the directions provided. The most difficult issues were obtaining and loading the licenses. ISS had support contacts that were extremely helpful and eventually I had the correct licenses and the software was operating. I was able to run scans against both UNIX and Microsoft systems. I learned how the different "levels" of scans were performed and how the various options were set within the software. I could reproduce the same results in the lab that we were receiving from customers.

Now that I understood how the tool worked, I could build a template for a response to the scans I was receiving. I took each section of a report I had received from a customer and separated it into an individual "response component."

Issue <XX>

SolarisFtpShadowRecovery: Solaris FTP server allows attackers to recover shadow file (CAN-2001-0421)

Additional Information *More Information*

The FTP server shipped with Sun Solaris versions 2.6, 7, and 8 is vulnerable to a buffer overflow in the glob() function. A remote attacker can overflow a buffer to cause the system to dump core. By using the "CWD~" command and the glob() function exploit, an attacker can cause parts of the shadow file, which includes sensitive information such as encrypted passwords, to be dumped to core. A local attacker can exploit this vulnerability to obtain passwords of other users from the shadow file and possibly gain elevated privileges on the system.

Response:
Sun has resolved this issue and it is included in the latest Recommended Sun Patch cluster. This has been tested but is not part of the standard Product installation.
Action:
Installation of the Patch Cluster can be scheduled at <Customer> convenience.

Figure 3 Response Component

Each response component contained a graphic image of the ISS vulnerability cut from a report, our response, and an action plan to resolve the issue. Once I performed testing and hardening described in the next section I could fill in the Response and action items. By placing keywords like <Customer> in brackets I could quickly customize the components and compile them into a document for the customer. This provided a response that mirrored the report the customer gave us and clearly identified the resolution and plan of action for each vulnerability identified in the report.

Nessus

The next tool I utilized was Nessus[5]. Nessus is an open source scanner tool that utilizes a client/server architecture. A server daemon running on a UNIX based platform actually performs the scans. The client is a graphical User interface to the server daemon. It can be run on a UNIX or Microsoft based system.

I chose to run a pre-compiled version of nessus available from Sunfreeware[6] (http://www.sunfreeware.com/). This website is sponsored by Sun and is part of the Sun Software, Information & Technology Exchange (Sun SITE www.sun.com/sunsite). It is an excellent source for compiled and un-compiled software for Solaris as well as links to open source software sites such as Nessus.

The compiled version typically lags behind the current version available from the open source site, but you don't have to set up a complete development environment if you use the compiled version, and that can save considerable time. The package also requires several toolkits and libraries that Sunfreeware had links to if you did not already have them. These included GTK tool kit for the Graphical interface, nmap for port scanning, ssl for secure communication and generation of certificates and the GNU gcc Compiler and associated libraries. The necessary components were all outlined at the Sunfreeware website.

The installation is a Solaris pkgadd and once that was performed the user can follow the directions from the http://www.nessus.org/demo website to configure for the first use. This included obtaining the latest set of plugins, creating a user, starting the nessusd daemon and Nessus, the user interface.

Once installed, Nessus is relatively intuitive to use. Each time the user must confirm that the latest set of plugins is installed. I was able to run several scans on a variety of system configurations. In comparison, both ISS and Nessus found virtually the same vulnerabilities.

Objective Opinion

Our company's Marketing Department had been receiving various vulnerability information from a variety of sources including our customers, myself, Support, Development and others. They wanted to obtain a third party analysis of our product for

vulnerabilities, so they hired the security consulting firm Akibia Inc (www.akibia.com) of Westboro, MA to review our product architecture and perform a series of scans on the systems. The consultants performed virtually the same scans as I did utilizing ISS scanner and Nessus. They also utilized a product called APP Detective.

The outcome was virtually the same, there were no surprises. The consultants found the same vulnerabilities that myself and others had identified through scanning and system examination. Even though there was no significant difference in what was found, there was considerable value in having the vulnerabilities independently confirmed by a third party. That was information Marketing and Sales could utilize with our customers to confirm our responses and action plans.

Responding

Table 1 lists the vulnerabilities we identified in our UNIX based products that could be addressed without significant impact on the application or standard environment.

Vulnerability	**Repair**
Apache revision 1.3.12	Recompile and package 1.3.29
OS Vulnerabilities	Recommended Cluster patch installation
NFS share	Restrict through dfstab
Open ports	Identify and document
Various OS Services	Disable via init scripts
Network Services	Disable unnecessary services via inetd.conf
Services ftp	Replace with sftp and provide script modifications
Services X	Disable X services

Table 1 Vulnerabilities

Apache

One of the biggest challenges was recompiling and integrating an updated version of Apache. The current running version, 1.3.13, was identified to contain significant vulnerabilities including CVE-2002-0392[7]. Apache is utilized on our systems as an interface for internal connectivity, not external connectivity to the internet .

Increasing presure from our customers and the security community made it imperative that a newer version be made available. I agreed to attempt to compile and test with our most critical sites the 1.3.29 version. Just this change alone would significantly improve the security of our product.

The recompilation was straightforward. I obtained the source from the Apache project website[8] and utilized the GNU developer toolkit provided with the Solaris Companion CD.

Because our critical customers were running at two versions of our product, which spanned over Solaris 8 and Solaris 9, I was required to create and test two packages for each version of Solaris.

The build procedure is well documented in the source download from Apache in the README and INSTALL documents[8.] Our build required all the modules to be included so I utilized the –enable-mods=all switch. After the compilation I integrated the components that had been customized by our developers including httpd.conf, apachectl, and the servlets.

After the build the integrated package was tested on lab systems by myself and Development and then tested in customer environments. No significant issues were encountered as a result of the newly compiled 1.3.29 Apache.

One important result of this effort was to bring a company wide awareness of our dependencies and responsibilities in regards to third party products such as Apache, Samba, and Linux and others. An initiative to identify and schedule updates of these products as part of our normal development cycle was implemented. A cost/benefit/risk analysis of third party components will now be performed when new features and functionalities are proposed. We also integrated into our Customer Support escalation process a method to for our support engineers request upgrades and patches to these dependent components prior to release in critical circumstances.

OS Patches

Our systems are qualified with the latest Solaris recommended cluster patch available at the beginning of each point release QA cycle. Not all customers upgraded with each point release, so many were far behind in system patches. Customers are requesting that specific patches or vulnerabilities be addressed that are corrected by Solaris OS patches. Cluster patches are released regularly by Sun[9] that include all the latest patches recommended by Sun as well as the necessary dependencies.

We have found that unless a critical issue needs addressing, the Solaris recommended cluster patch is the safest and most reliable way to keep a system up to date.
The cluster patch is incrementally built and automatically determines the proper order and dependencies. Once applied, subsequent patches that are already applied are skipped.

"Making the patching process simple, easy, and reliable"[10] is identified by the National Cyber Security Partnership (NCSP) as an important step in security improvement. In one of the earliest security issues addressed by my company, our latest software revision segregated the installation of the OS from our product. This allowed the latest cluster patch to be applied at the time of build and the system to be upgraded with subsequent patches with the same risk as applying any patch. We developed and documented procedures for the installation of the patches in our environment. The

procedure is straightforward and can be performed by customers or by our company's support teams.

OS Ports and Services

Shutting down unused ports and services is identified in several security hardening best practices documents such as Minimizing System Services from the Sans Security Essentials with CISSP[11] or Cert Security Improvement Module Practices # 4[12.]
We needed to identify and confirm which ports and services were required and under what circumstances. Because of the distributed nature of our application and site specific customizations, there was no hard rule on what was required and what was not.

Our initial approach was to identify those services that under most circumstances were not required and list them in a "Ports and Services Document". We would also provide audit services to our customers to identify specifically what additional ports and services, if any, could be shut down.

Customers that were very sensitive to security issues and had high security requirements required ports information so that they could not only shut them down if not required but could utilize smart switches with Access Controls Lists (ACLs) or firewalls to limit access to/from the system. We identified over 70 custom ports that our applications communicate on. Because some customers wanted to firewall the system we found it was also important to not just identify the ports and the corresponding applications, but to align them with the functional components of the system. For example, communication ports to/from the end user work station or management workstation needed to clearly identified so that if a fire wall separated them from the servers the customer would know which ports to allow through.

Documentation

As described in the Vulnerability Section above, we produced multiple documents on system vulnerabilities based on the scans and lab testing. These documents were formatted such that they could be easily modified and built to meet specific queries of internal and external resources. By combining the scanning information with the results of the testing and hardening, the response gave a complete picture of the issue to our customers. Additional documentation including hardening guides, Executive and Customer presentations, ports and services documentation and a Corporate Security Handbook was developed with information provided by the scan and lab work.

Knowledge Base Population

We began populating our internal knowledge base with security information based on the findings of the vulnerability scans and the hardening exercises. One of the major challenges of populating any knowledge base is to have a consistent method of organizing and setting keywords so that the items can be found when required. Below

is an example of a solution I placed into our knowledge base for disabling Simple Network Management Protocol (SNMP).

Goal: How do we disable snmpdx on Xproduct systems
Fact: Security
Fact: SOLARIS
Fact: snmp, snmpXdmid, snmpdx
Fact: CAN-1999-0517
Fact: VU#154976
Fact: SUN Security Bulletin 00178
Fact: Product Version 7.0
Note: A variety of snmp related vulnerabilities have been identified under Solaris on XXXX and XXXXX systems. SNMP daemon is not required in order for Xproduct to function in production .
Fix: To disable snmpdx
1. su to root
2. /etc/init.d/init.snmpdx stop
3. mv /etc/rc3.d/S76snmpdx /etc/rc3.d/DISABLED_S76snmpdx

Figure 4 Knowledge Base Example

The **Fact** Statements are analogous to keywords that the AI engine parses when a question is entered by the user. It was important to include a “CVE” or “CAN” or other standardized vulnerability classification into the knowledge base entry so that a user can search on the vulnerability ID and get a response to that vulnerability. Entering Facts like “Security” would produce a complete listing of security information and these could be limited with Keywords on the specific OS or product revision.

This solution allowed support, field and sales resources to examine the knowledge base when presented with specific security questions and as a centralized location to store the responses and resolutions.

The Results

A review and measurement of the goals listed in Section 1 undertaken by the GTS Security team reveals the following:

Goal: Develop and document a process to respond to Sales and Support questions on security issues.

By documenting and publishing the Security Queue Flow process, Sales, Support, Management and members the Security Team were able to understand how requests would be prioritized and processed. Senior management also understood their role and

under what circumstances they would be engaged. This significantly reduced confusion about where and how requests were handled and allowed us to measure the progress of each.

Utilizing the GTS Query Form as single point of entry into the Security Queue Process improved the quality of the requests we received and allowed management and the SME resources to track the requests.

This resulted in reduced time for responses to be sent. Prior to the above initiatives, queries were sent to individuals in various organizations. Responses could take days or be lost. After implementation of the Security Queue, an initial response and assignment was provided in less than one business day.

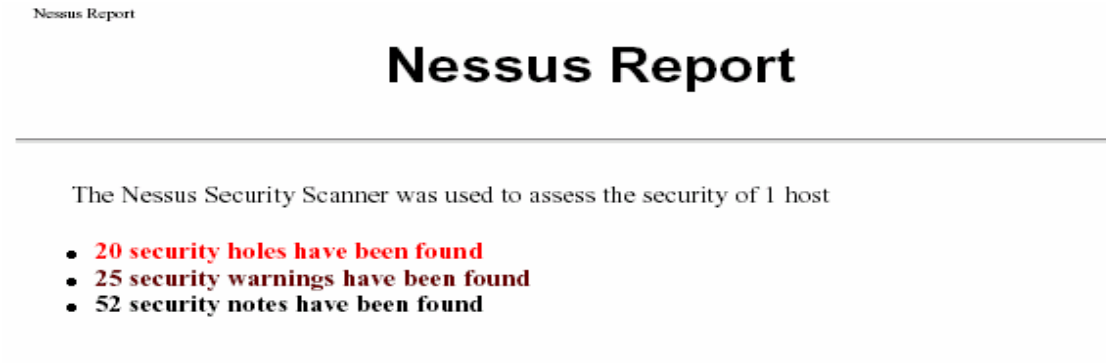
Goal: Verify customer requests such as scan reports or specific vulnerabilities.

By obtaining scanning tools and performing scans in the lab, we were able to validate the vulnerabilities in a controlled environment. We could evaluate whether the vulnerability was a direct result of our products and/ or a specific issue related to the customer environment.

Once validated, the specific vulnerabilities could be forwarded to appropriate resources including the SME, Development, Marketing or Support for a response. This response could be validated in the lab. The response could then be cataloged into a “Response Component” and/or the company knowledge base for quick access.

When I first started responding to vulnerability scans from customers, it took me over a week to confirm and validate the response with various resources. Now I can respond to many of them in less than one day. More importantly, other resources in the company can respond to the issues without even involving the SME. The templates and examples assist the Sales teams in responding to customers security questions and Requests for Comments/Proposals (RFCs, RFPs), which shortened the sales cycle when security issues arose.

Finally, having the vulnerability tools allows us to test and measure our progress in resolving specific vulnerability issues. Below is a Nessus scan summary I performed on an unhardened system.

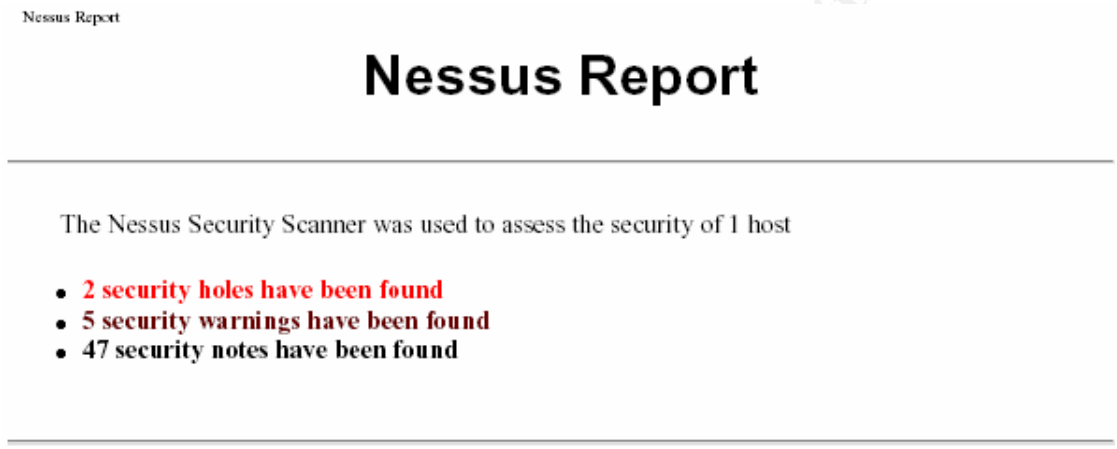

Nessus Report

Nessus Report

The Nessus Security Scanner was used to assess the security of 1 host

- **20 security holes have been found**
- **25 security warnings have been found**
- **52 security notes have been found**

After performing the recommended hardening steps the scan reported the following:


Nessus Report

Nessus Report

The Nessus Security Scanner was used to assess the security of 1 host

- **2 security holes have been found**
- **5 security warnings have been found**
- **47 security notes have been found**

This provided a relatively objective method to verify and measure our progress internally and with our customers.

Goal: Respond to specific hardening/customization engagements or requests

The Security Queue process, in conjunction with the test environment, allowed me to quickly evaluate the degree of effort that would be required for specific customization engagement requests. I could classify the categories of vulnerabilities, make system changes that removed or reduced the vulnerability, and then document and/or implement the appropriate changes on production systems.

Our team made significant changes in the OS Configuration for some customers. For example, removal of “R*Services or NIS+, both of which our application was dependent on, required testing and in some cases escalation to development for modifications to scripts and executables. The testing and hardening methodologies described in this document enable us to address and move forward with these requests in a timely manner.

Prior to the Security Queue Team, many operating system changes and configuration changes were considered “not supported”. By testing various hardening techniques in the lab I was able to illustrate to both our customers and our internal organizations that some changes could be performed without impact to the system.

We identified a scalable method to respond to OS vulnerabilities as well as the need for Marketing and Development to document a policy on the testing and application of patches and service packs on systems running our applications.

Utilizing the test systems as a basis we have been able meet 90%+ of our customers’ requested system hardening requirements. This reduced the probability that an attack would be successful.

Goal: Document security related issues and processes

Over 100 customers with specific security issues received documentation that was developed as part of this initiative

Documents that were developed as a result of our effort are:

- Security Queue Process Flow
- 2 Ports and protocols documents
- 25 ISS vulnerability responses
- 10 Hardening recommendations for specific systems
- Top 10 recommendations for securing our systems presentation
- 5 extensive engagement reports for specific customer customization

The knowledge base has been populated with our most critical vulnerabilities and the resolution. This effort continues and new entries are happening regularly.

Conclusion

Our Security Initiative directly contributed to making our products more secure by:

- Creating a method to quickly respond to security questions and issues
- Identifying System vulnerabilities
- Producing documentation and procedures to address system vulnerabilities
- Addressing system vulnerabilities through
    - Applying latest OS patches
    - Hardening practices
    - Open source upgrades
- Documentation of configuration and communication information for in securing the installed environment.

Our security initiatives contributed to many other tangible results:

- Increased corporate security awareness at all levels
- Introduction of security reviews into our development and QA cycles
- Establishment of a corporate initiative to review, consolidate and expand policies in regards to security
- Sales focus on our security initiatives and our security strengths
- Significant product enhancements and future security enhancements
- Introduction of the concept of a "layered" approach to security[13]
- Development of a Corporate Security Handbook.

Over the last several years, security issues have been gaining greater attention and resources throughout our company, as they are increasing in priority with our customers. The documentation of the process to handle the requests, the vulnerability scans, and the hardening testing this paper focuses on have played a significant role in increasing the security of our products and setting the direction for future enhancements and policies.

Prior to our initiatives, resources in our company were not aware of where to go to get a consistent answer on customer security concerns. Now they have a clear path.

In my opinion, the most important contribution of the GTS Security Team and I was that we acted as a catalyst for the exchange of information between organizations. We succeeded best when we were able to get various organizations to understand the often-contradictory positions they held (for example, access versus security).

A comparison of where we were when the GTS Security Team started and where we are now shows significant and measurable progress due to our activities. Our company now has the tools and direction to respond to current security threats and future ones as they develop.

References

1) Ober, David, GTS Query Outlook Form GTS, 2003

2) Cole, Eric Fossen, Jason Northcutt, Stephen Pomeranz, Hal Sans Security Essentials with CISSP CBK Version 2.1 Volume 1 SANS press April 2003 pg 841

3) Information Security, Georgia Institute of Technology ISS Internet Scanner, 2002 http://www.security.gatech.edu/information/iss_tutorial

4) Internet Security Systems, Internet Scanner http://www.iss.net/products_services/enterprise_protection/vulnerability_assessment/scanner_internet.php

5) Deraison, Renaud, The Nessus Project 1993-2004 www.nessus.org

6) Sun Freeware Project, Solaris 9 nessus-2.0.9 ftp://ftp.sunfreeware.com/pub/freeware/sparc/9/nessus-2.0.9-sol9-sparc-local.gz

7) Common Vulnerabilities and Exposures website CVE 2002-3092, 2002 http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0392

8) Apache Software Foundation Apache 1.3 INSTALL,README 1999-2004 http://archive.apache.org/dist/httpd/apache_1.3.29.tar.gz most recent version available at: http://httpd.apache.org/download.cgi

9) Sun Microsystems, Recommended Solaris Patch Clusters. Various by OS revision http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/patch-access

10) NCSP Task Force Report Improving Security Across The Software Development Lifecycle Executive Summary Pg 1, April 2004. http://www.cyberpartnership.org/SDLCFULL.pdf

11) Cole, Eric; Fossen; Jason, Northcutt; Stephen Pomeranz; Hal Sans Security Essentials with CISSP CBK Version 2.1 Volume 2. SANS Press, April 2003 Chapter 32 Minimizing System.

12) CERT Security Improvement Practices, Offer only essential network services and operating system services on the server host machine http://www.cert.org/security-improvement/practices/p068.html, 2001

13) Cole, Eric; Fossen, Jason; Northcutt, Stephen; Pomeranz, Hal Sans Security Essentials with CISSP CBK Version 2.1 Volume 1 SANS Press April 2003 Chapter 6