Global Information Assurance Certification Paper

SANS GIAC Practical Assignment
Version 1.4b
David C. Lyon

Methods for Securing a Multi-Platform Environment

1 Abstract

With limited resources in today's IT environment, managing systems from multiple vendors with multiple operating systems is a major challenge. In this case study, we examine one such environment and how the existing set of tools and processes were modified and expanded to provide better security for the systems and applications management function. By using mostly free software and custom scripting, older but stable legacy environments were successfully leveraged to continue to provide high availability and overall stability. The problem was that much of the environment was built on older and less secure protocols and methods. The solution had to look at the overall environment and had to provide a common framework to address systems management, applications management, systems monitoring, and remote management. The solution had to provide a more secure solution that would support those functions. We will identify the risks associated with the status quo and examine the protocols and methods chosen to provide better security. We will discuss the steps implemented to better secure the environment. Finally, we will present the outcome of this work and discuss some issues that were identified through the process.

It should be noted that some services were "off limits" and therefore, we were at the liberty of policy and settings determined by the responsible administrators of those services. In addition, we don't cover very basic hardening techniques such as disabling services and using ACLs to lock down directories and files.

2 Description of the Environment

The environment consisted of Compaq Alpha file servers, Alpha database and mail servers, IBM Citrix and file servers and Sun SPARC servers. A breakdown of servers and operating systems is provided below.

Operating System	Server	Purpose
OpenVMS	Alpha	DB/Mail
OpenVMS and Advanced Server	Alpha	File/Print
Tru64	Alpha	Mail/LDAP
Windows NT Terminal Server	IBM Pentium	Citrix
Windows NT/2000	IBM Pentium	File
Windows 2000	IBM Pentium	Application
Solaris	Sun SPARC	Database
Solaris	Sun SPARC	Application

Originally, the environment consisted mainly of Alpha servers but as the environment expanded, a common method for backups and monitoring was sought and developed. The author maintained most of the systems described here and collaborated with colleagues and project team members on security related issues.

2.1 Systems Management

A set of tools was used to provide some basic level of systems management across the disparate environment. Some of these tools ran as scheduled jobs and tasks and would report some vital statistics on a daily basis. Each platform was running the appropriate utility for the particular operating system being used.

- Account Management

The windows environment consisted of a central account store that was partially maintained by an automated process. Much of the account store was maintained manually but automated processes existed to ensure that all steps were completed that were necessary for account creation. Local Windows accounts were used very sparingly. In the OpenVMS environment, there existed an account base to provide for Macintosh file server access and it was maintained manually. In the UNIX environment, the Tru64 servers were not for general use but served specialized functions. The Solaris server environment included a manually maintained account base and provided access to the end user. One Solaris server functioned as an application server.

- Backup Process

Most servers in the environment used a common backup algorithm. All backup scripts reported backup status and the basic state of the system to a central server running OpenVMS. Some servers ran a script that copied backup sets to the central server where they could be backed up to tape. Some of the backup sets were rotated off-site as part of a disaster recovery plan.

- Local and Remote Access

Systems management functions were done primarily using Telnet, FTP and in a one case, IIS web administration. As the environment expanded, some Windows management was done over Windows Terminal Server and Citrix Metaframe. Remote access was generally done over Telnet and later, through Terminal Server and Citrix ICA clients.

- Patch Management

Patch management consisted of installing the latest service packs when possible and spending much time manually reviewing Solaris and Tru64 patches. Tru64 had a better patch analysis tool – dupatch.

- CGI

A set of CGIs was tightly integrated into the environment and provided systems monitoring, account management, application utilities and end-user listproc mail management. Access to the CGIs was controlled and logged.

2.2 Systems Monitoring

A custom application existed to monitor well known and custom services as well as application health. The application was scheduled to run at regular intervals throughout the day. Other scripts were deployed locally when appropriate. There was also a monitoring script that perused a central status database and made a daily report. This status database was updated each day by scripts running on OpenVMS, NT, Tru64 UNIX, and Solaris servers. They reported backup and system status by running a remote command on the server hosting the database.

2.3 Applications Management

There were several automated processes that existed to help manage specific application environments. These ranged from automated and manual processes to transfer files, to user interfaces to access data. There was also a process and environment that provided tight configuration control for the many scripts, web source and source code. It included a tool to deploy web content, scripts and programs using local and remote copy (rcp).

Several automated procedures and scripts existed to update databases, process reports and move them, and provide listproc mail management. There also existed a sophisticated Email activated reporting tool that was jointly maintained by the author and a customer, another Email activated tool used to update a status database, and a process to automatically move files to an out-of-state location.

3 Risk and Analysis

As indicated, because of the challenge presented at managing systems from multiple vendors, a common framework for basic management had been developed. It was first established for a single platform and it grew to accommodate other platforms over time. With that growth, the challenge was to provide secure automated monitoring, backups and regular patch management, and very basic intrusion monitoring for Windows servers. As the Internet grew, the risks of using insecure protocols to achieve some of these goals presented some not so obvious but nonetheless serious risks. Due to the lack of a DMZ, these risks were magnified. There was also pressure to set up data links

between hosts and provide easy access to reporting but security was generally not a primary concern. As awareness increased, the flaws in the methods and protocols were identified and targeted for hardening.

Much of the push for hardening was due to a Title V based grant project whose goal was to improve access at the participating institutions by increasing security expertise and awareness. It would achieve its goals by conducting vulnerability assessments, collaboration, development of policy and providing opportunities for security awareness training. Hardening the environment consisted of smaller projects, all contributing towards the long-term goals of the project. The project utilizes a measured approach lasting several years. By using the Nessus Security Scanner (http://www.nessus.org), project staff members were able to scan for a myriad of vulnerabilities. It was vital in helping identify some common risks and provided a means of mitigation. The author was actively involved in most aspects of the project.

3.1 Weak Protocols

Most system access and automated scripting was reliant on weak protocols such as Telnet and Remote Shell. In this section, we identify those protocols and the alternatives available.

3.1.1 Telnet and FTP

Telnet and FTP were very commonly used in the environment. The former provided general user access and the latter was frequently used by the user community and in scripts providing automation. There is the common misperception among some that providing credentials (user/password) makes these applications secure. It is true that the risk is less in a switched network but that doesn't necessarily deter the determined hacker and is not a substitute for encryption. ARP spoofing could be used to circumvent the switched environment.

A switched network could be prone to a man-in-the-middle attack. The malicious computer essentially goes between the target and destination and sniffs traffic along that communication path while still forwarding packets. In Sean Whalen's paper "An Introduction to ARP Spoofing" he states:

> When a MiM is performed, a malicious user inserts his computer between the communications path of two target computers. Sniffing can then be performed. The malicious computer will forward frames between the two target computers so communications are not interrupted (Whalen, p. 3).

Whalen indicates that this could be done by poisoning the ARP cache so that both computers associate the others IP address with the MAC address of the malicious computer, effectively making it the man-in-the-middle.

Of course, remote WAN access via Telnet is by far the greater risk here. One option that existed and was a logical choice was to use Secure Shell. OpenSSH (http://www.openssh.org) was already used in the organization and is open source. We would be installing SSH services on Windows, OpenVMS, and UNIX. The challenge would be to find workable solutions for non-UNIX environments. The choice for SSH also made sense in that clients that functioned similar to the telnet clients were already available or better yet, existing clients had added SSH functionality to newer versions.

- **Windows Access**

For the Windows side, encrypted remote access could be provided with the Citrix ICA client. Since Citrix was deployed in support of a large project coupled with the fact that ICA provided encryption, it made sense to use it. Windows NT Terminal Server 1.8, FR1 comes with SecureICA, providing high encryption (up to 128 bit key). SecureICA offers much better security than the basic security setting. It uses RSA's RC5 algorithm. The product literature states the following.

> A brute-force attack using a system capable of 100 MIPS would take 10,000,000,000,000,000 years to break SecureICA encryption at 128 bits. Even in the unlikely event someone succeeds, SecureICA would ensure that the viewer would see only meaningless screen commands and not the desired information (Citrix literature, p. 1).

Citrix ICA client is especially useful because it also maps local handheld drives as virtual disks, making file transfer from handheld to server very easy. For example, an administrator might need to get a log dump from a server and study it off-line. With the ICA client, it is simply a matter of drag and drop to the Window CE handheld storage card.

3.1.2 Remote Shell and Login

The common protocols RSH and RLOGIN were used to run remote commands from within scripts, to copy files, log in without a password, and deploy scripts that were maintained in a central library for the purposes of configuration control. In some cases, passwords were stored in plain text and transmitted across the network. In other cases, the trusted hosts mechanism was used. Both RSH and RLOGIN can be configured to use the hosts.equiv and rhosts files so that the command or login can occur without a password. While the former method is in no way optimal, it may be less risky (in a switched network) than the latter. In the latter method, an IP address could be spoofed, thus allowing an outside hacker a possible way in. The hacker could change the packet header (assuming he/she identified a trusting host IP) and be given trusted access. J. Suresh Kumar writes in PC Quest:

> The best method of preventing the IP spoofing problem is to install a filtering router that restricts the input to your external interface (known as an input filter) by not allowing a packet through if it has a source address from your internal network. In addition, the network operator should filter outgoing packets that have a source address different from his internal network in order to prevent a source IP spoofing attack originating from your site (Kumar, p. 1).

As with Telnet, the logical alternative to RSH/RLOGIN would be SSH. The end user community did not typically use RSH. It was used from within scripts to execute remote commands. Because automatic login was used with RSH, an alternative must offer the same functionality. SSH could be configured to use Public-key authentication. Because SSH is open source and supports Public-key authentication, it was the logical migration path.

3.1.3 Public-key Authentication

Public-key authentication is used so that a particular user on a client machine does not need to use a password when connecting to an SSH server. The user will use an SSH client to connect to the SSH server. This involves two keys. There is a private key that must be kept private and should only be available to the client user. The public key is placed on the server. The public/private keys are generated at the same time and are considered a pair. The client must generate a signature from the private key and provide this to the server. The internet-draft states the following:

> With this method, the possession of a private key serves as authentication. This method works by sending a signature created with a private key of the user. The server MUST check that the key is a valid authenticator for the user, and MUST check that the signature is valid. If both hold, the authentication request MUST be accepted; otherwise it MUST be rejected. (Note that the server MAY require additional authentications after successful authentication.) (Ylonen, p. 7)

Typical SSH client applications that were tested don't have a mechanism to send a password in batch mode. Public-key authentication was ideal in that no clear text password had to be stored.

3.2 Server Patches and Vulnerability Assessments

Because there was no systematic approach at keeping patches up to date, risk increased with every new CERT advisory (http://www.cert.org). Management of multiple operating system patches was a great challenge as all have unique tools for checking and updating patch levels. The method was to patch when there was time or if one was made aware of a severe vulnerability. There was no effective means utilized to check systems for vulnerabilities.

Automated patching methods were considered but can be very expensive and would not be practical unless one had to manage a large farm of like servers. With smaller collections of disparate servers, existing and freely available tools sufficed. Due to the fracturing within the organization, agreeing upon a standard for patch management was not going to happen soon. Patch notification and periodic installs of patch clusters would need to be key parts of a patch management plan.

3.3 Backups

On Windows servers, there was no effective backup of the security log. The risk here has to do with the logs being set to overwrite after a certain size. Also, if a hard drive crashed, the log could be forever gone. For forensics and legality, log files need to be backed up daily to provide traceability in the event of a formal incident.

In a one instance, there was a need for a secure backup (encrypted) across the network. The server was behind a firewall and backed up across the network. A decision was made to modify the commonly used Windows NT backup script (written in Perl) to copy the backup files using SSH secure copy.

3.4 IIS Administration

IIS requires vigilance in order to keep it patched and locked down properly. In one particular instance, it was installed and available as a management platform. When running NT4 server, there is no terminal server functionality and therefore, installing the IIS Admin scripts appears to be a good option. However, historically IIS out of the box is NOT secure and with the number of vulnerabilities reported, using it for system administration would be too great a risk. It would need to be de-installed.

3.5 CGI Form Input Filtering

While security was taken into account in the development of CGIs, the issue of form data filtering was not. This would be more of an issue on a UNIX or Windows server but nonetheless, still presents risk in an OpenVMS server environment, especially when passing arguments to some other program on a remote host. Data validation was provided where required by the CGI but it was not provided (other than very basic) for free-form inputs. An attacker could craft data in such a way to corrupt files or do some other damage. In the example below, we see how easy it might be in UNIX to do damage.

Form input: `` `rm *` ``

Perl CGI script code:

```
system("report.sh $web-input-file");
```

In the above example, "report.sh" could be a script that gets some system status information and formats it in HTML, depending on the first argument passed to it. In this case, whatever is between the back tics will be run, removing all non-hidden files in the directory where "report.sh" lives. The script will still run, in fact and depending how it was written, won't return anything since `rm *` will be null. Someone could conceivably issue a command to return information to the browser. Web servers should run as a non-privileged user.

3.6 Orphaned Accounts

While the adding of accounts was synchronized with an existing account store and partially automated, there was no automation for the removal of accounts when employment was terminated or consultants left. The risk is that these stagnant accounts could be easier to compromise and chances are, nobody would know. Maintaining and monitoring of accounts was left up to the account managers who were already over-taxed.

3.7 Custom Applications

A particular application that was triggered via Email to generate an Email response in the form of a report was at risk due to the type of validation required. Over time, the method could be figured out through educated guessing. Another issue was that incoming mail was not restricted to the local network. A better algorithm was needed that include some random attribute an added internal layer of security such as mail header validation.

Another custom Email driven application was used to log security issues and other project work. It used a product called "pmdf deliver" (http://www.process.com). The "pmdf deliver" used a rule file to determine whether a message would cause a script to execute. The rule can require a special string. What were lacking were other layered checks such as mail header and size restrictions. This would involve adding some parsing of the mail message headers. The other layers of security would improve the posture of the application.

A custom data transfer process was in development. Due to the lack of agreement on whether the remote server would push or pull sensitive data from the main server, a stalemate was reached. The perceived risk of the proposed solution was too great and therefore a compromise that met security requirements of the project had to be agreed upon. The mitigating solution was to provide a "middleman" server that would accept the data from the main server and allow the remote server to pull the data. Since the "middleman" server already existed, was not allowing general user logins, and was based on the highly secure OpenVMS operating system, both parties could come to agreement.

3.7.1 Clear Text Passwords

In many instances, it was determined necessary to have a stored clear text password for a service account, which in most cases would be a non-privileged account that was needed to provide automation of processes. While they were kept in locked down directories, there was still some risk of compromise due to the number of accounts that could override privilege. Again, Public-key authentication was the best solution to this (see section 3.1.3). While this was the preferred solution, there were applications that could not readily take advantage of it. In those cases, some form of file encryption was needed. Certainly, having the clear text password in a script was not acceptable. Hiding the details and the encrypted file was adding another layer of security.

- Remote Mail

The need to remotely read mail that contains systems status or problem reports is very important. The problem is that the POP protocol was often used in such a way that the clear text password went out across the wire. The mail client for Windows CE 2.2 did not support secure authentication or encryption. For those times when handheld devices were being used, a secure way to read mail was going to be required. Eudora for Palm OS was the choice because it provided the secure authentication and encryption. The remote Palm OS synchronization (over dial-up) was available for mail synchronization but it was slow and insecure and so Eudora became the logical choice.

3.7.2 Generic Accounts and Remote File Transfer

For one particular large project, a high risk existed in the way that files were transferred to a remote site. Since the clear text password and data had to route into the Internet, the risk was much greater. The other issue here is the fact that generic accounts were used and it was more difficult to trace who used the account. Without local logging of actual username, traceability was much more difficult and harder to prove.

Several alternative options were discussed. Would having a batch command that users could run enhance traceability? What about using a drag and drop method? Would a solution put us more at risk than before? These were questions that were discussed before arriving at a solution. It was decided that the best solution would be to have a file transfer listener process that would be driven by a fileshare. The fileshare would have an ACL that permitted only those with a specific role/group to put files for transfer. The listener could determine the user that placed the files and then could move the files, shielding account information from the user. All transactions would be logged. This was the only method (out of those discussed) that could ensure accurate logging. Secure Copy was again chosen to replace FTP as it was already in use and there was expertise available.

4 Steps to Secure the Environment

It should be noted that solutions to security related issues are not always optimal but sometimes are the most reasonable and provide a reasonable level of security and still permit business continuity. Dramatic changes in any environment will lead to a business slowdown and are unacceptable to the user community. In an ideal world, an organization would have a homogeneous environment. Many would have issues with the use of service accounts but the cost of NOT using them could be far greater than the cost of adding better security.

Most of the source code that had to be changed was located in a central repository and could be identified relatively quickly. Once changed, new versions were created and deployed to local and remote locations using a custom deployment tool where possible. Having a source code control system made the task much more doable.

4.1 Systems Management

In this section, we discuss the steps to secure the parts of the environment related to systems management functions.

4.1.1 Replacing Telnet and RSH

To build and deploy SSH on the disparate platforms was a challenge. A free version of SSH client was available for OpenVMS and it proved useful for a time but it only supported SSH version 1 which is known to have a design flaw. Process Software (http://www.process.com) has released a commercial grade product for OpenVMS that supports the more secure version 2 of the SSH protocol. We participated in the beta test and ended up purchasing the product. For Tru64 and Solaris, OpenSSH was chosen. It is freely available from http://www.openssh.org. Building these versions was time consuming because several other kits were needed. A large package, OpenSSL

was required but was freely available. We began the practice of disabling Telnet on dedicated servers and encouraging the use of SSH on general-user systems.

For Windows, two solutions were used. The first had already been deployed as part of another project. That solution was the Citrix ICA services and client. See section 4.1.2 for more information.

Not all Windows servers could run Citrix due to licensing and the need for the Terminal Server version of Windows NT 4. For all servers, we needed a secure means to update status and deploy scripts. The best free solution was the solution from Network Simplicity (http://www.networksimplicity.com), which is based on the Cygwin (http://www.cygwin.com) utilities. This distribution provided both the server and client pieces. It is much easier to setup and configure than are the Cygwin utilities. See section 4.2.1 for information on setting up authentication.

4.1.2 Remote Management

The Citrix ICA client is available for many systems. See http://www.citrix.com/download for a list of supported clients. This made it the hands-down choice for remote management. It also provided the encryption and therefore, remote access did not require a VPN connection due to the strong encryption provided by SecureICA (see section 3.1.1). In our case study, two hand-held devices were used for some level of remote management. The preferred device was an HP Jornada 820 (http://www.hp.com/jornada) running Window CE version 2.2.

The Citrix client worked very well on this device and supported the encryption level we required. With the Citrix ICA client, one could "terminal server" into a Windows server and have a virtual desktop. Because of the encryption provided, true remote access was provided whether from a coffee shop in New York City over a cell modem, a wireless connection at Starbucks, or from a landline dial-in at home. Of course, the option to SSH into a server and have a command line prompt was existed but for Windows servers, one can only do so much with the command line.

- **Palm OS**

  There was yet a smaller, Palm OS based handheld that was used. It doubled as a cell phone. The Samsung SPH-I300 proved very useful for systems monitoring and for urgent issues; very limited systems management. The Top Gun SSH client is available for Palm OS but it is a beta version and only supports version 1 of the protocol. More can be found at http://online.offshore.com.ai/~iang/TGssh.

  The other useful tool for PalmOS was Eudora (http://www.eudora.com). Eudora provides for a secure Email connection for both sending and receiving using the

POP protocol. Eudora has built in security functionality developed by Certicom Security (http://www.certicom.com). This functionality includes SSL and several authentication methods including CRAM-MD5 that avoid sending clear-text passwords.

To set up Eudora on Palm OS, one would enter the "Options" menu and then "Accounts". Click the "Security" tab and set the authentication to "best available" and security (SSL) to "Alternate Port" for the incoming connection. For the outgoing connection, you will need to set authentication as "best available" if your palm dials in off-network and define security (SSL) as "Alternate Port". These settings are probably typical but depend on the mail server configuration.

The strip utility (http://www.zetetic.net/products.html) also provided for password storage in an encrypted format for Palm OS.

4.1.3 Patch Management

For all systems in this multi-vendor network, a practice was implemented to make sure the servers were kept as up to date as possible. With UNIX servers, patch kits were generally installed on a periodic basis, depending on urgency. If a vulnerability that a patch addressed was determined to be high risk, then an effort was made to install the patch sooner. Non-critical patches were generally done once per week, usually on Friday evening, the optimal time for the organization. Also, security related changes were logged to a database.

The overall patching practice and method used is described below.

- Join CERT announce list (http://www.cert.org/contact_cert/certmaillist.html)
- Join SANS announce lists (http://www.sans.org/newsletters/)
- Join Microsoft announce list (http://www.microsoft.com/security/security_bulletins/decision.asp)
- Review the SANS Top 20 list (http://www.sans.org/top20/) and verify we are guarding against the attacks listed in that list.
- Run Nessus scanner periodically (http://www.nessus.org) to check for vulnerabilities.

Windows Servers

- **Run Windows Update (**http://windowsupdate.com**)**

This application will survey your system and attempt to make a determination whether hotfixes are needed. It doesn't update for all components and that is one reason to be on the Microsoft announce list.

- **Run Hfnetchk (**http://www.microsoft.com **- search for "hfnetchk")**

  This is a command line alternative method at identifying hotfixes your server might need. This may be better at identifying hotfixes Windows Update does not.

- **Run Baseline Security Analyzer (**http://www.microsoft.com/ **- search for "Baseline Security Analyzer")**

  Baseline Security Analyzer is a GUI vulnerability scanner that produces easy to read reports.

- **Use QCHAIN (**http://support.microsoft.com/default.aspx?scid=KB;EN-US;296861&id=KB;EN-US;296861&sd=tech **)**

  The qchain program available from Microsoft can be used to apply multiple hotfixes with one reboot. Applying multiple hotfixes individually without reboot after each can result in a newer version of a file getting overwritten by a patch that contained the older version.

- **Ghost Backups**

  Periodically, Windows servers were backed up using Ghost (http://www.symantec.com). Ghost was used to do a complete image backup. A database is used to log all patch updates and system changes, providing a recovery audit trail if a system drive (and spare) ever failed or were found to be corrupt.

Solaris Servers

In the past, reviewing the patch CD was very tedious, partially due to all of the patch dependencies. Sun has released the smpatch tool (see http://docs.sun.com/db/doc/816-0211/6m6nc677b?a=view) to provide "Windows Update" functionality for Solaris. This tool has proven useful and reduces the time needed to identify and install patches. There are some issues with it, however. Smpatch has problems running in single-user mode and was not useful for applying the patches. Some custom scripts were created to help install the patches recommended by smpatch that were in the recommended patch kit from Sun. The resulting procedure is described below.

- Run “smpatch analyze” to get a list of patches
- Download Sun’s recommended patch kit
- Run custom scripts to prepare for patching
- Run script that calls patchadd in single user to apply only relevant patches (from smpatch list)

Tru64 UNIX Servers

Subscribe to the Tru64 UNIX patch list (See http://www.hp.com/united-states/subscribe/gateway/?jumpid=go/subscribe-gate1). Run the utility “dupatch” against the latest aggregate patch kit (after Email notification) and select the option to do the pre-installation check. From this, a list that is appropriate for the server can be generated and installed using dupatch.

This method is not perfect because of dependency issues that can be confusing.

4.1.4 Backups

There was a new mandate to store nearly every day’s backup off-site. This required changes to backup scripts that had been storing multiple sets onto one tape. While this was an important change, it was far more work for the operations staff who had to maintain the tape logs.

As discussed in section 3.3, there was also a need to provide an alternative to backing up a remote server to a Windows fileshare. This was an issue on one server in particular, due to a firewall and the nature of the data being backed up. The general windows backup script was modified to allow for secure backup using SSH with Public-key authentication. Very few ports were open on the firewall, port 22 being one of them.

Not all clients are created equal. Early on in the deployment, we discovered that our scp client was often hanging on the connection. When the client from SSH Communications Security (http://www.ssh.com) was deployed, our problems ended and we have had very stable backups as a result.

Another key change to the script was to have it back up the Windows Event Security log. This script was deployed on almost all of the Windows servers in this case study. Because the file was always open and we were not using a commercial backup application that could handle this, we added some code to dump the event log and then back it up to the server. The “dumpel” utility (available from Microsoft) was used from within Perl. See the example below.

```
`dumpel -l security -m security -d 1 -c –f \\backup\\zip\\security.log`;
```

With limited staff managing a hodge-podge of servers, it wasn't practical to monitor the security logs on a regular basis. Code was added to the same backup script to extract suspect log entries and format them in HTML and mail them to the systems administrator. While certainly not intended to function as a commercial intrusion detection system, this did help identify patterns of abuse against each server it was run on.

The following command (from within Perl) collected the raw data.

```
my(@log)=`dumpel -l security -m security -d 1 -c -e 529 539`;
```

The above command dumps records from the security log for the past day for errors 529 and 539 (logon failure and account lock out). The Perl script parses the log array and converts the format to a readable color coded format in HTML and then Emails the log to a local address.

4.2 Script Automation

As previously noted, scripts that needed to move files or run commands remotely relied on FTP and/or remote shell/copy. In the case of OpenVMS, running ssh or scp from within a Perl script proved more difficult than was anticipated. The Perl libraries available were not deemed usable on OpenVMS and therefore, custom subroutines were written and placed in a Perl library. By using a subroutine call, the details of the secure copy itself were shielded from the code, allowing for a much easier upgrade when the time came to upgrade the free SSH client with a commercial version from Process Software (http://www.process.com). A script that was used to deploy files from a central OpenVMS server was modified to use Secure Copy for OpenVMS. See the relevant code below.

```
$ ! DCL code to transfer file to VMS/non-VMS host using SCP2 from
$ ! Process Software
$ !
$ dft=f$ext(f$len(destination)-1,1,destination)
$ !
$ ! Determine if transferring to non-vms and set qualifiers
$ if destination .nes. "" .and. dft .nes. ":" .and. dft .nes. "]"
$ then
$   destination=destination + "/"
$   vmq="/translate_vms"
$ else
$   vmq="/vms"
$ endif
$ destination=destination+file
$ scp2 'vmq'  "''file'"  "''user'@''host'::''destination'"
```

For Bourne shell and Windows scripts, no library was required. It was a relatively simple matter to replace the rsh/ftp commands with the SSH equivalents. The most difficult part was getting the right number of quotes when running a remote command that required arguments as the sample below illustrates.

```
#!/bin/sh
#
# Run the dcl command procedure "dbinput" on the remote host, passing
# arguments to update status
#
/usr/local/bin/ssh -2 -p 22 -l service host.domain @dbinput  \
   "\"$fn\"" "\"$boot|$succ|$date|$err\""
```

4.2.1 Authentication

Because of the need for authentication in batch mode, Public-key authentication needed to be set up between the hosts that would participate. For the backup/status script, this was not terribly complicated since there was one main status server. The complication arose when trying to get Public-key authentication working initially. Once the right combination of commands to generate key pairs was found, they were documented for each operating system. Because the SSH software for OpenVMS was an early release, it also compounded the problem until a more stable version was installed.

For those instances where Public-key authentication was not feasible, the passwords were kept in an encrypted file. A library was written that was used to manage the file. The added layer of security mitigated some of the risk of having to store passwords.

- **Key Generation**

Because of the multiple procedures for key generation on the various platforms, we include the procedures for the benefit of the reader that might be challenged with setting up something similar.

- **Windows (SSH Secure Shell Communications Client)**
  - Log into the account that will be used.
  - Run "Secure Shell Client"
  - Select Edit->Settings
  - Select "Keys"
  - Click "Generate new"

- Click "Next"
- Select DSA 1024 bit key
- Click "Next"
- Provide a filename and comment
- Answer passphrase question
- Click "Finish"
- Click "Configure..." and OK (three times)
- Close the application and click "Yes" to save settings
- Open a cmd tool
- cd to "Application Data\SSH"
- scp the newly created ".pub" file to the server's [.ssh2] directory

- **OpenVMS**

  - Log in into the account that will be used and enter:

```
$ create/directory [.ssh2]
$ set sec/prot=(g,w) ssh2.dir
$ multinet sshkeygen /ssh2 /keytype=dsa /bits=1024
$ set default [.ssh2]
$ create indentification.
idkey id_dsa_1024_a [ctrl-z]
```

- **UNIX**

  - Log in into the account that will be used and enter:

```
% mkdir .ssh
% chmod -R g-rwx,o-rwx ../.ssh
% cd .ssh
% ssh-keygen -t dsa -b 1024
% ssh-keygen -e -f id_dsa.pub > ~user/id.pub
```

- **OpenVMS Server**

Use the following procedure for each client public key you are transferring.

Note: pulling the public key files from the remote client servers was preferred if the server was running OpenVMS due to VMS record format issues. The record format must remain “STREAM_LF”.

- Log in into the account that will be used and enter:

```
$ create/directory [.ssh2]
$ set sec/prot=(g,w) ssh2.dir
$ set default [.ssh2]
$ scp2 "user@client::id.pub" client-name.pub
```

- Add "key client-name.pub" to the end of []authorization.

4.3 CGI Hardening

Most CGIs required a layer of authentication so in most cases, we had the authenticated username and were tracking activity in a log file. Since our CGIs were created within a common framework, it was clear in each script where to do the hardening/form data validation. Most CGIs used a common library to parse form data. Some filtering was done at that level already. It was also important to audit any code that contained system calls to run other commands. If a user caused a validation error, a mail message was automatically sent to notify a support person that either someone was having trouble understanding what data to enter into a field or someone was doing something bad.

An example of field validation is provided below.

```
for ("name","phone","title") {
  #
  # If a field has data, check whether it contains any disallowed
  # characters.
  #
  if ($dh{$_} && $dh{$_} =~ /[^\w\s\.\-\(\)]/) {
    &Error_Handler("Breakin detection");
    last;
  }
}
```

4.4 Account Maintenance

A central account database feeding off of a reliable source already existed and was updated automatically. The logical choice was to synchronize actual account bases with this database. A Perl script was developed that would run weekly to flag any orphaned accounts. Since all permanent accounts were tracked using a unique identifier, we were able to effectively determine accounts that didn't belong. Also, the database kept track of expiration dates for accounts and the script utilized this by setting the expiration date for NT accounts. There was also provision to track special accounts such as those for consultants. NT account deletions could be done automatically but due to the business

nature of the accounts, the decision was left up to the account administrator who would get the weekly HTML report of orphaned and special accounts.

The script also tracked Sun/Solaris accounts but would not take action. These accounts were few and were maintained added and removed manually. The procedure provided the means to help keep those in sync with the database.

CGI and account maintenance scripts were also updated to keep track of account adds/deletes and maintain tables that were mapped against the database, keeping track with the unique identifier. This would help prevent orphaned accounts from being created. The VMS accounts used for Apple fileshares were retired because a decision was made to not offer this service anymore.

4.5 Application Hardening

- **Data Transfer**

There were several application scripts that warranted hardening. A particular need was for secure data transfer between two very different platforms running completely different applications. One system ran a job to export data to a CSV file and the other converted the CSV to useable data for it's proprietary application. This needed to happen automatically at least once per day to keep data in sync.

As discussed in section 3.7, the issue arose around pushing versus pulling of the data. One problem was that the remote, dedicated server was behind a firewall and was not running SSH server. Adjusting firewall rules could take time and was prone to problems. The mitigating solution of providing a "middleman" server to accept data from the main server and serve it to the remote server was implemented. This authentication was accomplished using Public-key authentication (discussed in section 3.1.3). The "middleman" server held the public keys for the other two servers. This proved to be very stable and secure.

- **File Transfer with Generic Accounts**

In section 3.7.2, we discussed the need to transfer files to a remote site that housed database servers. The responsible administrators at the remote site would provide generic accounts, generally one per department. Users needing to move files would be given the credentials.

As previously stated, this is sub optimal due to the inability to do accurate tracing. The solution to create a file transfer listener was implemented. Perl was used to

write the listener. One critical challenge had to do with determining the ownership of files that were posted to the listener fileshare. Users that were a member of a specific group were given access to drag files to the share. The listener was always running and would process the directory. Our version of Perl had no clear way to determine a file owner in Windows. The solution to this was to use the "cmd2k.exe" program from the Windows 2000 distribution. The sample code is listed below.

```
my(@ownst)=`cmd2k.exe /c dir /q "$filename"`;
split(' ',$ownst[5]);
my($own)=$_[3];
```

The file transfer method was still based on FTP, which for obvious reasons, stated in this document, was unacceptable. There are Perl libraries for SSH but a determination was made that it would be faster to just do a background command (`scp source dest`) to use an SCP client to do the transfer. One particular SSH library requires several other libraries and there would be no guarantee that it would work. We already knew the SCP client worked. The solution here was to create a new subroutine that used the Secure Copy (scp) command and reliably checked for success and failure.

- **Email Driven Applications**

Two Email driven applications were discussed in section 3.7. In both of these instances, simple clear text authentication was used to cause scripts to execute. In the application that was used for reporting, the author teamed up with a developer/customer and came up with a better method for authentication. In both applications, other layers of security were added such as requiring specific values for mail headers. Also mail to the reporting server was blocked at the firewall.

5 Outcome and Summary

This environment is a part of a larger heterogeneous network. One person cannot dictate standards but all have to come into agreement on issues surrounding weak protocols. For example, there are issues with old client PCs that may only support LM (LAN Manager) authentication, a known weak method.

There are issues of cost. In the case of OpenVMS, we had no choice but to buy a commercial version of SSH. One could probably make an argument that in this case, commercial might be better due to having technical support. In fact, there were problems with the free SSH client that were not an issue in the commercial version.

There is one as yet unsolved technical issue. The particular secure copy (scp) client used for one application was chosen because it can send the password if in a batch

stream. This might be required, depending on our ability to use Public-key authentication on a remote host not under our direct control. The client we used is not stable and is prone to hanging. The best solution would be to use the stable client already in use for other applications and configure Public-key authentication.

- **Summary**

  When coupled with knowledge from hands-on training, the subject of this case study provided a great learning experience. Much was also learned from hands-on and a lot of time with a search engine. The actions taken were concrete and were designed to produce a clear outcome. Some of the benefits as a result of the hardening of the environment are included below.

  - Minimization of saved passwords
  - Encrypted authentication
  - Encrypted data on the network
  - Account management under control
  - Monitoring and management scripts using secure methods
  - Highly secure OpenVMS leveraged
  - CGI form data filters hardened with break-in notification
  - Secure remote management
  - Much better patch management
  - Windows intrusion report
  - Vulnerable services disabled
  - Vulnerability assessments are done

  The challenge of hardening a multi-platform environment was successful using a common security practice and relying on good protocols (SSH, ICA), tight configuration control, and good record keeping. The end result has been a stable, more secure environment.

6 References

1. Whalen Sean. “An Introduction to ARP Spoofing.” Apr. 2001. URL: http://www.rootsecure.net/content/downloads/pdf_downloads/arp_spoofing_intro.pdf (29 June 2003).

2. Citrix literature. “SecureICA Services.” 1999. Citrix. URL: http://www.praxium.com/pdfs/SecureICA.pdf (28 June 2003).

3. Harwood, Ted. Windows NT Terminal Server and Citrix Metaframe. Indianapolis: New Riders, 1999. 88.

4. Nemeth, Evi, Garth Snyder, Scott Seebass, and Trent Hein. UNIX System Administration Handbook, Second Edition. Upper Saddle River, NJ: Prentice Hall, 1995. 716.

5. Kumar, J. Suresh. “IP Spoofing.” 5 May 2003. URL: http://www.pcquest.com/content/topstories/spy/103050505.asp (27 June 2003).

6. Ylonen T, T Kivinen, M Saarinen, T Rinne, and S Lehtinen. “SSH Authentication Protocol.” 20 Sep 2002. URL: http://www.ietf.org/internet-drafts/draft-ietf-secsh-userauth-16.txt (28 June 2003).

7. Arce, Iván. “SSH1 CRC-32 compensation attack detector vulnerability.” 8 Feb. 2001. URL: http://www.securityfocus.com/archive/1/161448 (29 June 2003).

8. Process Software. “SSH Solutions on OpenVMS – Technical Overview.” June 2003. URL: http://www.process.com/tcpip/SSH_SPD_Finalb2.pdf (29 June 2003).

9. Microsoft. “How to Install Multiple Windows Updates or Hotfixes with Only One Reboot.” 28 May 2003. Microsoft TechNet. URL: http://support.microsoft.com/default.aspx?scid=KB;EN-US;296861&id=KB;EN-US;296861&sd=tech (29 June 2003).

10. Microsoft. “Automating Detection of Logon Failures In a Windows NT Domain.” 14 May 2003. Microsoft TechNet. URL: http://support.microsoft.com/default.aspx?scid=kb%3Ben-us%3B171148 (29 June 2003).

11. Savill John. “How can I prevent the OS from storing LAN Manager (LM) hashes in Active Directory (AD) and the SAM?” 8 Oct. 2001. John Savill’s Windows NT/2000 FAQ. URL: http://www.ntfaq.com/Articles/Index.cfm?ArticleID=22817 (28 June 2003).

Last Updated: September 22nd, 2017

Upcoming Training

SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS New York SEC401^	New York, NY	Sep 25, 2017 - Sep 30, 2017	Community SANS
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, Denmark	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Sacramento SEC401	Sacramento, CA	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS DFIR Prague Summit & Training 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
SANS Phoenix-Mesa 2017	Mesa, AZ	Oct 09, 2017 - Oct 14, 2017	Live Event
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
SANS Tysons Corner Fall 2017	McLean, VA	Oct 14, 2017 - Oct 21, 2017	Live Event
CCB Private SEC401 Oct 17	Brussels, Belgium	Oct 16, 2017 - Oct 21, 2017	
SANS Tokyo Autumn 2017	Tokyo, Japan	Oct 16, 2017 - Oct 28, 2017	Live Event
SANS vLive - SEC401: Security Essentials Bootcamp Style	SEC401 - 201710,	Oct 23, 2017 - Nov 29, 2017	vLive
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC401: Security Essentials Bootcamp Style	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS Gulf Region 2017	Dubai, United Arab Emirates	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Miami 2017	Miami, FL	Nov 06, 2017 - Nov 11, 2017	Live Event
Community SANS Vancouver SEC401^	Vancouver, BC	Nov 06, 2017 - Nov 11, 2017	Community SANS
Community SANS Colorado Springs SEC401~	Colorado Springs, CO	Nov 06, 2017 - Nov 11, 2017	Community SANS
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
SANS Sydney 2017	Sydney, Australia	Nov 13, 2017 - Nov 25, 2017	Live Event
SANS London November 2017	London, United Kingdom	Nov 27, 2017 - Dec 02, 2017	Live Event
Community SANS St. Louis SEC401	St Louis, MO	Nov 27, 2017 - Dec 02, 2017	Community SANS
Community SANS Portland SEC401	Portland, OR	Nov 27, 2017 - Dec 02, 2017	Community SANS
SANS San Francisco Winter 2017	San Francisco, CA	Nov 27, 2017 - Dec 02, 2017	Live Event
SANS Khobar 2017	Khobar, Saudi Arabia	Dec 02, 2017 - Dec 07, 2017	Live Event
Community SANS Ottawa SEC401	Ottawa, ON	Dec 04, 2017 - Dec 09, 2017	Community SANS
SANS Austin Winter 2017	Austin, TX	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Munich December 2017	Munich, Germany	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Bangalore 2017	Bangalore, India	Dec 11, 2017 - Dec 16, 2017	Live Event
SANS vLive - SEC401: Security Essentials Bootcamp Style	SEC401 - 201712,	Dec 11, 2017 - Jan 24, 2018	vLive