Global Information Assurance Certification Paper

Evading Network Security Devices Utilizing Secure Shell

SANS Security Essentials (GSEC) Practical Assignment

Version 1.4b – Option 1

Completed By: Wesley D. Brown

Completion Date: November 9, 2003

TABLE OF CONTENTS

1. Abstract

This document has been written with the intent of exposing readers to the mindset of an administrator, user and attacker. Understanding the mindset and thought process of these roles is essential to the security administrator's role in information security. This document provides examples that introduce the thinking process of the above listed roles in an effort to provide security professionals with a better understanding of individuals with malicious intent and their goals. It is my hope that any person who reads this paper will not only ask what a specific protocol or technology can do for them, but will also ask what that product can do to them when reviewing technologies.

2. Introduction

Since information security has become one of the main focuses within the information technology field, the information technology professional's role has changed significantly. No longer are administrator's and engineer's worried about just transferring data, but now they must do it only after authentication and a secure path has been established. Outdated protocols like telnet, ftp and rlogin are no longer an acceptable means of performing day-to-day operations. Organizations now rely upon products like Secure Shell (SSH) to perform day to day tasks in a secure manner. These products all have encryption, authentication and integrity mechanisms built into them. These mechanisms ensure that the connection has not been exposed to attacks like eavesdropping or "man in the middle" attacks. As much as these technologies may help an organization defend against attackers, they can also cause a substantial amount of harm to a network. Rogues users and attackers can utilize security and encryption technologies to achieve their goals in the same way that administrators do.

This could be the subject of much debate, but the one thing that cannot be debated is that the potential for users and attackers to manipulate encryption protocols is prevalent. It is also relatively simple to implement. I personally would rather an attacker get into my network without utilizing encryption techniques to bypass network security. At least then administrators would be able to see the data going through a network then not have the ability at all. At least then if an attacker has compromised a machine, an analyst can detect the event and assess the amount of damage incurred once the intrusion has been detected. Providing the organization has put the proper processes and tools in place to detect such an attack. If an encryption mechanism such as the SSH protocol is utilized by a hacker or rogue user, the content of the traffic can not be validated by an organizations firewall, filtering router or intrusion detection system (IDS) policy to ensure the traffic meets the organizations network security policy. The

Internet Protocol (IP) address and port will be known, but not the data inside of the packet. And as most sites are configured to prevent or block only proven malicious data that is in violation of a rule or entry, the chance that this data will not violate a policy is great.

Within this document I will provide insight as to how an individual with malicious intent may manipulate industry standard security technologies to evade network security devices. Some common security technologies are Secure Socket Layer (SSL), Virtual Private Network (VPN) software and Secure Shell (SSH) software. SSH is the only protocol that will be used within this document though.

I will also discuss techniques an administrator may utilize to maintain a level of control over their network with the hope of being able to detect or prevent this technique from being utilized. Actual solutions would be dependant upon the environment and infrastructure in place.

3. Evasion by Secure Shell

3.1 Introduction to the Secure Shell Protocol

"SSH provides support for secure remote login, secure file transfer, and secure TCP/IP and X11 forwarding. It can automatically encrypt, authenticate, and compress transmitted data."

--Internet Engineering Task Force (IETF) SSH Working Group

By design, Secure Shell (SSH) is a protocol that provides a secure mechanism for security professionals to efficiently perform functions such as remote system administration, network communications and file transfer. The protocol has built within it the ability to encrypt data, authenticate users, ensure data integrity and improve performance by compressing the data to be transmitted.

Depending upon the SSH implementation that your organization utilizes, it more then likely has the capability of supporting Advanced Encryption System (AES), Triple Data Encryption Standard (3DES) and Blowfish. It may also have the ability to negotiate a Data Encryption Standard (DES) connection if all the previously listed fail to negotiate with the remote side. However, DES is considered outdated because of the weak encryption that it utilizes. Some recent operating systems may not even have DES support built in for this reason.

The protocol also has built in mechanisms for user authentication. SSH may authenticate users by password, Kerberos, RSA public-key, certificates or host-based client-authentication. An organization can utilize single or multiple authentication mechanisms in conjunction with one another

SSH also has the ability to compress data, which could be a huge benefit for a user with a slow Wide Area Network (WAN) connection or limited bandwidth, and also has the requirement to encrypt the data and authenticate users. SSH has multiple levels of compression, which is dependant upon the SSH implementation utilized by a site. SSH Communications Security has stated on their website that their implementation of SSH is capable of compressing a text file up to fifty percent.

The SSH protocol utilizes a client/server architecture. This means that an SSH server will either accept or reject incoming connections from SSH clients based upon the server's configuration. There is a number of SSH implementations available to fit any type of environment would need. Some of the common operating systems that have SSH applications available for them are Apple, Linux, Unix and Windows.

Given the different SSH implementations available, there is some basic terminologies that one should be familiar with. The following information was derived from O'Reilly's book "SSH The Secure Shell, The Definitive Guide".

SSH: A generic term referring to SSH protocols or software products

SSH-1: The SSH protocol, Version 1. When this abbreviation is seen, the Secure Shell implementation being discussed is that of the IETF's SECSH working group.

SSH-2: The SSH protocol, Version 2. This is also the implementation from the IETF's SECSH working group.

SSH1: This is the release of SSH (version 1) from SSH Communications Security Inc.

SSH2: This is SSH Communications release of Secure Shell (version 2).

ssh: A client program included in SSH. When referring to the client version, ssh will be in lower case followed by the version.

OpenSSH: This is the secure shell release from the Open BSD project. Built within this release of the secure shell protocol is support for both version one and two. This secure shell implementation is open source.

The Secure Shell protocol is always being enhanced to provide even more features and functionality. The IETF is continually striving to make the protocol more secure, more robust and simple to implement. They have defined the following goals for the Secure Shell Protocol (SSH):

- Provide strong security against cryptanalysis and protocol attacks.

- Work reasonably well without a global key management or certificate infrastructure.
- Utilize existing certificate infrastructures when available.
- Made easy to deploy and put into use.
- Minimum or no manual interaction from users.
- Reasonably clean and simple to implement.

3.2 Variations of SSH

Within this section, different implementations of SSH will be introduced as well as some of the features and characteristics of each of the implementations as listed by the respective developers.

- As stated on the developers website "PuTTY is a free implementation of Telnet and SSH for Win32 platforms, along with an xterm terminal emulator".

  The following information was compiled from the developer's website. PuTTY also supports SSH2, but the default SSH protocol utilized is SSH1. This application also can support public key authentication, both RSA and DSA, in SSH2. The major downside to PuTTY is that it only runs on Win32 platforms (Windows 95, 98, NT, 2000, Me and XP). Eventually it may be ported to Unix and Linux platforms, but it is currently not available. More information on PuTTY is available at the following location http://www.chiark.greenend.org.uk/~sgtatham/putty/.

- According to the developer "TTSSH is a free SSH client for Windows. It is implemented as an extension DLL for TeraTerm Pro. Teraterm Pro is a superb *free* terminal emulator/telnet client for Windows, and its source is available."

  TTSSH adds SSH capabilities to Teraterm Pro without sacrificing any of Teraterm's existing functionality. TTSSH supports multiple encryption types, port forwarding and compression. TTSSH does have some significant downfalls which will undoubtedly drive the industry away from it. This extension, and TeraTerm Pro, is only available for Windows platforms. Furthermore, it does not support SSH2. More information and the application can be found at http://hp.vector.co.jp/authors/VA002416/teraterm.html. More information and the TTSSH extension can be found at http://www.zip.com.au/~roca/ttssh.html.

- OpenSSH is probably one of the most well known and utilized implementations of SSH throughout the world. The following passage is taken directly from the OpenSSH website "OpenSSH is a **FREE** version of the SSH protocol suite of network connectivity tools that increasing

numbers of people on the Internet are coming to rely on. Many users of telnet, rlogin, ftp, and other such programs might not realize that their password is transmitted across the Internet unencrypted, but it is. OpenSSH encrypts all traffic (including passwords) to effectively eliminate eavesdropping, connection hijacking, and other network-level attacks. Additionally, OpenSSH provides a myriad of secure tunneling capabilities, as well as a variety of authentication methods."

The developers boast that OpenSSH has the following features: open source, freeware, strong encryption, X11 forwarding, port forwarding, strong authentication mechanisms, agent forwarding, interoperability, full SFTP support for SSH1 and SSH2 as well as data compression. To download or inquire further about the OpenSSH product, detailed information is located at http://www.openssh.com/.

- The next implementation of the SSH protocol that will be covered is from the company F-Secure. According to the company's website their SSH product is Federal Information Processing Standards (FIPS) compliant and contains components certified by the National Institute of Standards and Technology (NIST). In addition, their SSH protocol is also being standardized by the IETF as the only certified SSH implementation.

  This implementation supports most platforms (including Windows, Linux and Unix) which is a huge positive for multi platform environments. The main downside is the fact that the product is not open source and must be purchased. More information is available on this product at http://www.f-secure.com/.

- The following application is "simple, free, secure TCP and UDP tunnel program." The product developer's acknowledge on their website that their product is not able to compete with products such as ssh, SSL or FreeS/WAN. But after some research, I decided that their application was definitely worth mentioning because of the capabilities the product has to offer. This product is Zebedee (**Z**lib compression, **B**lowfish encryption and **D**iffie-Hellman key agreement). Zebedee's developers boast that their product "is a simple program to establish an encrypted, compressed "tunnel" for TCP/IP or UDP data transfer between two systems. This allows traffic such as telnet, ftp and X to be protected from snooping as well as potentially gaining performance over low-bandwidth networks by using compression."

  This application can run in either server mode or client mode. Another interesting thing about Zebedee is that it not only offers TCP connections, but also offers UDP connections. The application also has the ability to function under Linux, Unix or Windows. Zebedee also has the capability for the server to initiate the connection with the client. Some additional

configuration would be required on the host. This additional configuration is referred to as “listen-mode”. Technical information can be found on the developer’s website. The only downfall to this application is that it only utilizes the Blowfish encryption algorithm and the Diffie-Hellman key agreement. But this issue would only be relevant if an individual did not want to use the default encryption and authentication mechanism, or if someone wanted to integrate this into an environment with other security technologies. Complete documentation and the product are available at http://www.winton.org.uk/zebedee/.

- The final SSH variation is the company SSH’s Secure Shell implementation. They cover most platforms (Windows, Linux, AIX and Unix) and of course technical support is provided as part of the purchase price.

  The down side to this product is that it is not freeware, therefore costing money to implement a solution that could be done at no initial cost to your organization. More information can be found at the company’s website http://www.ssh.com.

- FreeSSH is more of a service provider, not an actual application. The FreeSSH site is trying to keep track of all the different SSH implementations available. This is a great list to review different SSH implementations before selecting one to deploy into your environment. The site does not list reviews, but does provide a consolidated location to find links to different implementations. For more information, go to their website located at http://www.freessh.org/.

3.3 Practical uses for SSH – An administrator’s perspective

So what does this protocol provide to the information security professional? SSH is a very robust, secure and simple protocol designed to securely perform everyday functions. The SSH protocol allows the security professional to encrypt data, such as user names and passwords, prior to sending that data across the network. SSH makes eavesdropping essentially useless, session hijacking, password sniffing and other network level attacks. As anyone can see, SSH is a powerful tool that can make any administrator’s job a lot more efficient and secure.

In addition to being able to securely perform functions across a network, authenticate users and improve performance; it can also be used to secure other protocols that do not have security mechanisms built into the protocol. Within SSH there is a function called port forwarding, also known as tunneling. Port forwarding, or tunneling, allows TCP data to be passed through the secure tunnel established by secure shell.

With port forwarding a server authenticates a user and establishes a secure tunnel with the client. The negotiation process will include authenticating the user by passwords, key's or certificates. When a user is authenticated the encryption method will then be established. Once the connection is made basic insecure protocols such as receiving email via the Post Office Protocol (POP) can be transmitted through an encrypted tunnel. It should be also mentioned that both the client and server configurations must agree to port forward in order for this feature to work.

The tunneling, or port forwarding, feature makes SSH an invaluable tool for information technology professionals, and it can be implemented at no cost. Utilizing this feature, an administrator can set a secure tunnel up for any type of data deemed necessary. This would of course need to be dealt with on a case-by-case basis. Having a large deployment of SSH clients wouldn't be very administrator friendly, so as one can see there are some concerns in respect to scalability.

3.4 Practical uses for SSH – A rogue users perspective

As useful as SSH may be for an administrator, it can be even more useful for a rogue user. A rogue user's intent is to bypass the security mechanisms put into place. The reasoning behind this could range from email privacy, communications privacy to hiding malicious criminal activity form network administrators. To bypass the entire network security infrastructure a user simply has to connect to an SSH server outside of their respective LAN. Once connected to that server a user has full access to the entire Internet via an encrypted tunnel. Once that encrypted connection is made a user could launch attacks, view unauthorized material or utilize an organizations network for criminal activities.

In order to accomplish this, the user must install an SSH client application on their computer. Applications such as PuTTY, TTSSH, ZeBeDee and OpenSSH are examples of freeware encryption applications. These applications are all relatively simply to install and configure. Extensive documentation on basic and advanced configuration is also available on the Internet. In order to connect to an SSH server, the user must acquire the configuration for that server and input the data into the client application. Once the correct configuration is applied to the client the user only needs to connect to the SSH server to have completely secure, unmonitored communications on the Internet.

There is no possible way to monitor the data contained inside the packet or block that user's data if in violation of an organizations information security policy because it is encrypted. Once connected to the SSH server, the user has full access to and from the LAN. Any security implementations and measures put

into place to validate the data that the user is either sending or receiving are useless because the data being passed is completely encrypted.

The rogue user now has bypassed every security measure that an organization has implemented. Without the data being verified by the firewall policy and without an intrusion detection system capable of monitoring it, that user's computer can now become a backdoor for any hacker to obtain access to a network. Once this happens the game is over, the hacker now "owns your organization". The attacker now has the ability to launch attacks, compromise other machines, deface websites, databases or documents, gain access to company proprietary information or even utilize your networks resources for their own benefits. The options are only limited by the imagination of the attacker.

The description above was just one example of how a user might use SSH to bypass network security devices. There are other ways that SSH could be used to bypass a security infrastructure. The objective of this section was to identify how a secure tool used by security professionals to aide in their job can also be utilized by a user to bypass the security implementations in place. Furthermore, this can be accomplished without ever being detected by a company's information security team.

The rogue user generally has one goal, to have total access to the Internet without having their traffic validated or checked by the organizations information technology policies or staff. While doing this may be considered harmless, every security professional knows the ramifications that this could cause on a network. In the following section, you will be shown how an attacker could use secure shell for their purposes.

3.5 Practical uses for SSH – A hackers perspective

A hacker may utilize many different security technologies and protocols to bypass a networks security infrastructure. The protocol that will be demonstrated within this section is the secure shell protocol. SSH could be used to establish a completely secure backdoor into any network of their choosing. The attacker would "simply" need to compromise a machine and install SSH on the computer once compromised. This section will demonstrate a technique of how an attacker could manipulate the secure shell protocol for malicious purposes.

An attacker could write a freeware or shareware program, such as an entertainment application that a user may want to utilize, and within that application insert a Trojan program that installs SSH (client or server, depending upon the attacker's preference) onto the host. The key to this approach is to insert a Trojan into an application that an unsuspecting user would be likely to download and install. The ideal application would be a computer game. Around the holidays, there are many games that are usually sent around via email that

have a holiday theme. This would be the ideal transportation mechanism. The thought behind this is that if a user is downloading a game, the last thing they will do is ask the organizations administrator to assist them on the game installation. Users frequently download these applications and install them without thinking twice!

Most Trojans would of course be detected by an anti-virus application such as Norton Anti-virus. The attacker would need to ensure that the trojan is written and installed in a method that will not trigger a virus signature to alert the user of suspicious activity. The ideal way to test this for the attacker would be of course to test it in a controlled environment. A controlled environment being one in which there is no chance the Trojan could make its way to the Internet. To accomplish this all an attacker would have to do is unplug their WAN connection. If a leak were to occur, it could potentially indicate the intentions of the attacker to security organizations world wide. This would allow those security organizations time to make a detection mechanism available and therefore making it less useful to the attacker.

While the user is installing the freeware application, SSH (client or server, depending upon the deployment method) would be installed as well. While installing SSH the attacker should also pre-configure the application according to their specific deployment implementation. The attacker would most likely tunnel the SSH connection through a well known port, such as the Hyper Text Transfer Protocol (HTTP, port 80) or the Secure Socket Layer (SSL, port 443) in order for the traffic to appear as thought it were valid and not be blocked by the firewall. This configuration is crucial for the traffic to appear valid and avoid being detected by an intrusion detection system (IDS), blocked by a firewall or filtering router.

The client should also be configured to automatically start the SSH service upon the operating system loading. This of course is dependant upon the operating system. If the attacker were to be thorough they would code the Trojan in a manner as to detect which operating system it would be installing upon. Once it detected which operating system, it could then install and reconfigure the system to start the application upon the system startup. This would make the application even more useful to the attacker because it would be available for any platform.

In order for the backdoor to be transparent to users and to administrators, the attacker should pre-configure the server or client to auto connect upon the service starting. The crucial point of this is that the connection initialization must be made from the infected machine to the attacker's machine. It is common knowledge that industry best practice for a firewall's configuration is to not accept incoming Transmission Control Protocol (TCP) connections that were not requested from within the network. Also worth mentioning, is that the majority of firewalls (and even some routers) have the capability to detect if a connection was requested. The attacker does have options to solve this problem though.

The attacker may use a feature within some SSH applications referred to as “remote forwarding”. Once the remote forwarding feature is utilized, the SSH server’s role changes. The SSH server becomes the application client and listening side. The SSH client becomes the application server and connecting side. This completely reverses the role each plays in the default implementation of SSH. In summary, this feature enables the attacker to either install an SSH client or server on a machine and also pre-configure that host to initialize the connection to the host specified by the attacker.

As I have demonstrated within this section, the only things an attacker needs to utilize the SSH protocol to bypass an organizations security mechanisms are a WAN connection and open port on the firewall or filtering router. If the attacker can detect that through network mapping and reconnaissance then they can manipulate the SSH protocol to their advantage.

Once the circuit has been established the attacker has complete access to the network to either utilize the SSH backdoor that has been established or to install another back door and then remove the SSH backdoor or disable it until needed again. The SSH backdoor provides the attacker with a completely secure circuit into and out of a network without any possible way for a network based security device to validate the data or to detect that a backdoor exists. This has the potential to be an enormous issue for the information security industry; one that I believe should be looked at more in depth by every organization.

There are many more features that are available for SSH and for network security devices. The point that I am making, is that no matter what network or host level control mechanism is attempted, this weakness will still exist in some manner. The SSH protocol, and many others, can be manipulated in anyway a hacker sees fit. Most SSH implementations are open source and can be easily modified. There is nothing to stop an attacker from modifying the original standard open source SSH implementation to an application that fits an attacker’s specific need.

A technical detail that an attacker would want to consider is to not use an industry standard certificate or authentication mechanism. If utilizing authentication features, the potential for an administrator to capture the authentication mechanism, once they detect the host has been compromised, and utilize that certificate or password to decrypt and monitor the communications taking place could be devastating to the attacker. A proxy, or man in the middle attack, could be used against the attacker to collect evidence which could be used to convict the attacker. This technique is of course illegal, unless the proper channels have been taken by the organization and law enforcement agencies.

An attacker should also be aware that the authentication, encryption and key exchange do not have to be an industry standard. From an attacker's standpoint, the ideal solution would be to develop their own authentication and encryption method and then modify the SSH implementation to utilize that. An attacker could also pre-configure a symmetrical, non-negotiable key. When a key negotiation takes place, it has the potential to be detected by an Intrusion Detection System, alerting an analyst that suspicious behavior is occurring. By pre-configuring the key to be symmetrical, the attacker may also now write a proprietary encryption algorithm. This could be either detrimental or beneficial, depending upon the proficiency of the attacker. The main thought one should gain from this paragraph is that SSH can be modified in many ways to fit a need on a case-by-case basis.

An experienced attacker could also code in multiple symmetrical keys and rotate between them with a metric that would randomly switch to a different symmetrical key. This technique would be somewhat similar to the way the military uses "frequency hopping" for their radio communications and also in some wireless Local Area Network (LAN) implementations.

Frequency hopping is a technique in which the frequency of operation for a given system or piece of equipment is changed at a random time to a random frequency. Every piece of equipment within that system would have the same metric for generation therefore staying on the same frequency. The reasoning behind this is that once a enemy found the frequency of operation, the enemy would have to then find the correct key before a frequency change occurred again, thereby reducing the chances of eavesdropping.

Modifications may need to be written to their manipulated implementation on a network by network basis. But as I have shown, with proper reconnaissance, secure technologies and protocols can be manipulated to provide attackers with a very powerful tool.

A point worth mentioning within this section is that an attacker is not worried about being detected. After all, even if an attacker were detected by an Intrusion Detection System, or other network security device, no one could read the data inside. And if no one can read the data being sent and received then it would make prosecuting the attacker significantly harder. The attacker has only a few topics with which to be concerned with. Some of those concerns are maintaining control of who connects to and from him and that the encryption algorithm and key being utilized cannot be detected, captured or used by others. This can be accomplished utilizing any numbers of options or manipulations. No matter what the solution or bypass an administrator may implement, there will always be a hacker that can find a way to bypass and achieve their goals. It is not a matter of if, just a matter of time until they do so.

4. Control Measures

Although there are many reasons to be alarmed with the information provided within this document, there are control measures an organization can take. None of these control measures will leave an organization completely safe, but they do provide a level of control against exploitation. The control mechanism could be either network or host based. The solution is dependant upon the environment and weakness that lie within. It seems as though a network based solution would be much harder to implement then a host based, since it is rather hard to look at encrypted data if one does not have the key. Therefore, mostly host-based solutions will be examined within this section.

By running scans on your network for the secure shell service you can verify which computers are utilizing the protocol. This method is only useful if there is a detailed configuration management system in place. If the computer that is running the service is not documented within the configuration management process, then it is definitely worth further investigation. This control method could easily be bypassed by an attacker with a bit of forward thinking (redirecting traffic through port 80 or port 443), but I believe it is a measure that must be utilized from a defense in depth approach.

An additional method that could be utilized is one in which the data collection is automated. This could be accomplished through a script being executed upon a user logging into a domain or upon some other type of action or time. The key to the script being a success would be that every service that the computer is configured to run should be running at the time the script is executed. The script would run a command that would allow the administrator to have the ports listening or processes being ran on that computer written to a log file of some sort. This script would then ideally redirect the output or transfer that file to a central log server. Every effort should be made to prevent that file from being tampered with. It is guaranteed that an attacker would try to prevent actions or files of this nature in order to avoid getting caught and having their backdoor closed. A cron job (scheduled task in Unix) could then be made to execute daily, or weekly. The cron job would execute the script so that it would run a search for the information specified within that script. As mentioned above, that information could be for ports (sockets) associated with secure communication protocols or system services commonly associated with secure communication processes running upon system initialization. There is any number of options an administrator could devise to make this possible. Once an administrator has this written, all they would be required to do is have the report sent to them. Although initial setup would take some effort, it is well worth the effort. Especially when considering the consequences.

Standard user systems should be profiled (configuration management) upon deployment. The idea behind profiling is that user systems rarely change after deployment, aside from system patches. It should be relatively easy to identify

when unauthorized applications are installed on the system. There are a number of tools available to inform an administrator when something or someone has modified a baseline system.

Proper user and file system permissions can also be utilized to control a service being installed or utilized. If the user does not have permission to install an application or enable a service on a system then the chances of them installing an application or initializing a service is drastically reduced. That user would then have to circumvent the permissions in place in order to utilize a service or install an application. This is a good indication that the user, once identified, is probably up to more then trying to browse the Internet.

There are host based security applications that could also be utilized. If a host based firewall, intrusion detection system or Transmission Control Protocol (TCP) Wrapper were to be installed in a restrictive posture then it would be much more difficult for an attacker to profile which ports are open on both the firewall and on the hosts. This practice is of course the very nature of the defense in depth concept.

Commercial vendors, such as Internet Security Systems, are also becoming aware of attackers utilizing secure technologies and protocols to bypass security devices. They are currently working on solutions to this problem. When talking to an ISS Engineer, he stated they had the ability to read and inspect SSL traffic once the certificate had been obtained in a lab environment. This is of course would be very easy for an attacker to bypass, because the attacker should not allow the certificate to be made available. It is however a start and it is only a matter of time until a practical solution is developed that will at least make it harder for rogue users or attackers to utilize encryption techniques for their benefits without at least being detected.

There is one network-based measure of control that I believe may be possible for some protocols and technologies. Some encryption protocols and technologies encrypt in a manner that leaves Internet Protocol (IP) protocols. These protocols specify what type of IP is occurring. Some examples of IP protocols that may be seen if a tunnel or encryption is taking place is protocol 47 (Generic Routing Encapsulation/GRE), protocol 50 (Encapsulation Security Payload/ESP), protocol 51 (Authentication Header/AH) as well as protocol115 (Layer 2 Tunneling Protocol/L2TP). There are possibly other indications that a tunnel or encryption is taking place. At least with this mechanism, an analyst could detect that it is occurring and capture the internal Internet Protocol (IP) address for further investigation. This IP protocol may or may not be different for each security technology, so individual applications would need to be investigated in order to produce signatures. Once the research is done though, an administrator could write custom signatures relating that specific IP protocol back to a specific application and then investigate that machine for the service that the alert was produced for. The administrator should instinctively take the position that there is

always a solution. It may not be an easy solution, one easily implemented or the ideal solution but there is more then likely at least one solution to help aide in their duties.

Every control measure in this section has the potential to stop attackers, but a savvy attacker could have also easily worked around each solution. The purpose of this section was to introduce a method of thinking, a method of thinking that every hacker has. This is the ability to solve problems quickly and dynamically. Every environment is different and the solution should fit the environment based upon the different variables that are introduced within that environment. An information technology professional must be able to identify the problem or potential issue and then engineer a solution based on extensive research, a granular understanding of the problem, industry standard best practices, maximum return for the minimum investment and their respective organizations requirements.

5. Conclusion

There are a limitless number of issues that could be introduced within this paper. The discussion of them is not within the scope of this paper. Within the confines of this paper I have referenced a secure protocol intended to assist information technology professionals perform their duties and how an individual could manipulate that protocol to bypass network security. In addition to bypassing, this also provides a user or attacker a completely secure tunnel into or out of your network without the possibility of being caught without control mechanisms. It is my hope that the mindset that has been shown in this paper can be utilized by administrators in order to understand a protocols benefits and drawbacks. Without understanding pros and cons of technologies, an individual's ability to provide a strong policy and network security infrastructure is significantly disabled. So the next time you are reviewing a technology ask yourself what it can do for you, and how can it be used against you!

6. References

6.1 Barnett, D., Silverman R., SSH The Secure Shell The Definitive Guide. Sebastopol, CA O'Reilly & Associates, 2001.

6.2 Sommerfield, B., "Secure Shell (secsh)". June 20, 2003. http://www.ietf.org/html.charters/secsh-charter.html.

6.3 Tatham, S., "PuTTY: A Free Win32 Telnet/SSH Client". September 24, 2003. http://www.chiark.greenend.org.uk/~sgtatham/putty/.

6.4 Teranishi, T., "Tera Term (Pro)". August 9, 1999.

http://hp.vector.co.jp/authors/VA002416/teraterm.html.

6.5 O'Callahan, R., "TTSSH: An SSH extension to TeraTerm". March 21, 2001. http://www.zip.com.au/~roca/ttssh.html.

6.6 "OpenSSH" http://www.openssh.com.

6.7 "F-Secure SSH" http://www.f-secure.com.

6.8 "FreeSSH". June 30, 2003. http://www.freessh.org.

6.9 "SSH" http://www.ssh.com.

6.10 " Federal Information Processing Standards" http://csrc.nist.gov/publications/fips/.

6.11 "Zebedee: Secure IP Tunnel" http://www.winton.org.uk/zebedee/.

6.12 "Internet Security Systems" http://www.iss.net .

6.13 "Frequency-Hopping Spread Spectrum" http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci525695,00.html.

6.14 "IP Packet" http://pw1.netcom.com/~jsnader/pg.pdf

Last Updated: July 28th, 2017

Upcoming Training

SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style	Virginia Beach, VA	Aug 21, 2017 - Aug 26, 2017	vLive
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
Community SANS Pasadena SEC401 @ NASA	Pasadena, CA	Aug 23, 2017 - Aug 30, 2017	Community SANS
Mentor Session - SEC401	Minneapolis, MN	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
Mentor Session - SEC401	Edmonton, AB	Sep 06, 2017 - Oct 18, 2017	Mentor
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Mentor Session - SEC401	Ventura, CA	Sep 11, 2017 - Oct 12, 2017	Mentor
Community SANS Albany SEC401	Albany, NY	Sep 11, 2017 - Sep 16, 2017	Community SANS
Community SANS Dallas SEC401	Dallas, TX	Sep 18, 2017 - Sep 23, 2017	Community SANS
Community SANS Columbia SEC401	Columbia, MD	Sep 18, 2017 - Sep 23, 2017	Community SANS
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, Denmark	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Boise SEC401	Boise, ID	Sep 25, 2017 - Sep 30, 2017	Community SANS
Baltimore Fall 2017 - SEC401: Security Essentials Bootcamp Style	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
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 London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
Community SANS Charleston SEC401	Charleston, SC	Oct 02, 2017 - Oct 07, 2017	Community SANS
Community SANS Sacramento SEC401	Sacramento, CA	Oct 02, 2017 - Oct 07, 2017	Community SANS
Mentor Session - SEC401	Arlington, VA	Oct 04, 2017 - Nov 15, 2017	Mentor
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Indianapolis SEC401	Indianapolis, IN	Oct 09, 2017 - Oct 14, 2017	Community SANS