Global Information Assurance Certification Paper

SSH and Intrusion Detection
Heather M. Larrieu
GSEC v1.3 Practical

Abstract

Widespread use of the SSH protocol greatly reduces the risk of remote computer access by encoding the transmission of clear text usernames and passwords. Prior to the use of SSH, packet sniffing, which allows malicious users to watch for the login process in the clear text packet traffic on a network segment, was an easy method for a malicious user to gain unauthorized access to a machine. Unfortunately, use of SSH might allow a malicious user to bypass intrusion detection systems because of its encrypting of the data payload and its ability to tunnel protocols. This paper outlines the role and issues with the use of the SSH protocol, types and methods of intrusion detection, and proposes techniques and an architecture for an intrusion detection system that uses the SSH daemon as a sensor.

Introduction

The introduction of the SSH protocol as a replacement for legacy remote access tools like Telnet and rsh has greatly decreased the risks associated with remote computer access and administration. Unfortunately, it has accordingly increased the potential for a malicious user to circumvent the best practices of computer security analysts by allowing some unauthorized use to go unnoted by intrusion detection systems that rely on clear text traffic analysis.

SSH Overview

There are many benefits to using SSH for typical remote interactions such as terminal access, file transfer, and batch commands. For example, it provides not only security, confidentiality, and data integrity via its encryption mechanisms, but also the secure use of legacy applications (such as the X11 protocol) via its tunneling capabilities.

The original remote login tools, which include rsh, rlogin, rcp, and Telnet, transmit data including user authentication data in clear text. This enables a malicious user to steal usernames and passwords by putting a sniffer on a shared network segment and watching for the login process. A sniffer is a program and/or device that monitors data traveling over a network. A sniffer can be inserted almost anywhere on the network and be virtually undetectable [1]. The introduction of switched networks makes packet sniffing more difficult since all of the network traffic on a segment is not broadcast to all of the devices on that segment, but is not a solution to the problem of sniffing. As Stephen Sipes notes, “The development of switched networks was driven by the need for more bandwidth, not for the need of more secure networks. Indeed, investigation

reveals that several methods are available to sniff switched networks. [2]”

SSH provides two secure authentication methods that protect the login process from network-based sniffers: standard password and key based; in the latter standard public/private key mechanisms are employed. To use key based authentication, a user puts in a file on the remote host her public key which is used in conjunction with private key stored on the connecting host during the authentication process. If the keys form a valid pair, the user is logged on. Generally, to get access to the private key, a user would be prompted for the passphrase that protects the key. It is possible, however, for a user to have a private key without a passphrase. If a user does not have a passphrase on her key, a cracker with access to the private key file could have the user’s complete access to the remote system.

SSH protects the entire communication session in addition to the user authentication process using encryption techniques. During the initial phase of the SSH connection setup, the client and server exchange packets containing fields which specify the algorithm and key to be used to encrypt the session. Interestingly, the RFC for the SSH protocol specifies that some protocol header fields are encrypted in addition to the data payload. These headers are the packet length, padding length, and the padding fields [7].

After initially connecting to a host, SSH can verify that any subsequent connections are made to that same host, or it can indicate that the host has changed based on the public key that the remote host presents. This prevents some of the so-called “man-in-the-middle” attacks by which a machine impersonates the intended destination host. In such an attack, the malicious host intercepts all of the source host’s data, and it can either respond as the destination host or it can store the traffic locally while relaying the communication or a modified version of the communication to the actual intended destination host. These “man-in-the-middle” scenarios represent attacks on both communication privacy and integrity.

Another benefit of using SSH is the ability to tunnel arbitrary TCP/IP connections that allows previously insecure communications to be used in a secure fashion. This is commonly used for X11 connections, and can also be used for virtual private networking.

Because with SSH the entire data stream is encoded using strong encryption schemes, getting useful information out of the traffic collected by the packet sniffer would require a large investment of time and effort on the part of the would-be data thief, but as David Rankin noted “what difference does it make that it might take 100 years to crack a message encrypted with a 512-bit key when it might only take 10 minutes to hack into the system and steal the key which allows you to decrypt the message in seconds?[3]"

Therein lies part of the problem with the use of SSH from the point of view of a computer security analyst. The machines from which users want to connect to the network necessarily lie outside of that network, and hence probably outside of a professionally maintained business-quality network. In particular, such a remote machine could be in a user's home, dorm room, library, friend's house or any of a myriad of public access points. If the machine is located in a home with a broadband "always-on" connection, experience suggests that in all likelihood the machine has been or will be compromised in some fashion. Such a compromise might arise from a virus, Trojan, or some other malware, which could then record keystrokes looking for juicy information like passwords or credit card numbers.

While no one has exact statistics, according to an article in the Pioneer Press earlier this year, of the approximately 100 million Americans connected to the Internet, only about 50 percent use anti-virus software, and only 5 to 10 percent run firewalls. The Pioneer Press also reports computer security incident reports have doubled over the period from 2000-2001 with the CERT at Carnegie Mellon reporting 52,000 incidents. The Nimda virus, which infected 2.2 million computers in 24 hours, counts as a single incident [4]. It these statistics are representative of reality, it is well within the realm of possibility that any user's password, passphrase, and private key are all easy targets. So while SSH solved the problem of clear text authentication and provides a greater level of security for TCP/IP based services, it has also introduced new issues, including the possibility that a malicious user might employ SSH to elude intrusion detection systems.

Intrusion Detection Systems

An intrusion detection system is a system that attempts to discover, alert, and possibly respond to an instance of undesired access. An intrusion can be defined as the attempted use of a system without authorization, an attempt on the part of a legitimate user to abuse or increase her privileges on a system without authorization, or an attempt to deny access to a computer system or application. It is important to note that an intrusion is an intrusion whether or not the attempt is successful. There are a wide variety of intrusion detection systems currently available or being researched. These include network based, host based, hybrid, and honeypots or honeynets.

A network-based intrusion detection system usually consists of a passive sensor machine which has a network card configured in promiscuous mode, enabling it to collect all of the network traffic that passes along its segment of the network. Network-based intrusion detection systems are generally easier to deploy than other intrusion detection systems, because they can monitor many hosts from a single location. Not having to install the detection system on many machines concentrates the administrative tasks required to maintain the system to a minimum set of machines. Since Network-based intrusion detection

systems are limited to detecting the traffic that is on their network segment, administrators must take care in their deployment plans to provide comprehensive coverage appropriate for their site's particular needs.

Host-based systems run on the host they are monitoring. In general, they watch the application and system logs for attacks. In addition to monitoring log files, host-based agents might monitor accesses and changes to critical system files and user privilege [8]. Host-based intrusion detection can also include monitoring of network traffic at the host along with monitoring of critical system files. System files can be monitored by keeping a record of checksums for installed files and periodically verifying the recorded value with the value reported by the file on disk. While not in the system proposed in this paper, host-based intrusion detection systems are included here for completeness, and should play a role in any comprehensive intrusion detection capability.

Hybrid intrusion detection systems combine the functionality of network and host-based systems using both traffic and log files for data. These systems usually consist of distributed sensors that monitor a host or a network segment, and report alerts back to a central monitoring console.

Honeypots and honeynets take a different tack than standard intrusion detection systems. A honeypot "is a resource who's [sic] value is being probed, attacked, or compromised [5]". Generally, a honeypot is a non-production system placed in a carefully monitored location accessible to the open Internet. Honeypots can display different levels of service depending on the intent of their manager, but should always be configured such that the compromise of the system or applications running on the honeypot cannot be used as the source of an intrusion into another system. This means that the honeypot should reside behind a firewall configured to contain traffic leaving the honeypot. Honeypots include full operating systems which actually perform no business function but which run modified kernel modules that are able to monitor and record all activity on a system unbeknownst to an attacker. In another variation, they might execute on a well-known port a simple state machine which simulates expected initial connection conditions so activity can be recorded and an attack analyzed. The use of honeypots gives the ability "to track attacker attempts at entry and respond before they come across a vulnerability we are susceptible to [6]". Honeynets are a virtual or physical networks of honeypots, although some groups, namely The Honeynet Project, specify that honeynets be for research purposes instead of being components in an intrusion detection system.

No matter what type of intrusion detection system is deployed, all basically use two techniques for detecting an intrusion. These techniques are to watch for known signatures and to watch for anomalies. Most systems use a combination. The search for known signatures or anomalies could potentially take place at every level of a protocol stack. To illustrate this detection capability at all levels of a communication, an abbreviated OSI protocol model follows, and examples

with the most common protocol suite, the TCP/IP, are presented.

Application
Transport
Internet
Network
Physical layer

Starting from the lowest and working up, the two detection techniques could be employed to detect intrusions at every level.

Detection at the physical layer would require detailed knowledge of electronics, but conceivably a model of the electrical patterns could be created as a basis for anomaly detection. The introduction of a new network card, a new piece of hardware, or a cable modification should be detectable. While out of the scope of this discussion, it might be relevant to extremely secured networks.

The same could be said of the network layer where the Ethernet frames could be examined for the basis of anomaly detection. A process that monitors a network segment for new MAC addresses could be an element of anomaly detection. Presumably the number of detects that would be available at this level is limited at best, but potentially unauthorized hosts that connected to the segment would be detected.

The Internet layer deals with the IP protocol. At this level, the packet contains multiple elements that could be used in signature or anomaly detection. For anomaly detection, a baseline of standard network traffic would have to be collected and analyzed for a statistical representation of the data. Any packets deviating from this statistical model should be reported for further investigation. Knowledge or signature based detection would include looking at the settings for each field in the IP packet headers. IP header fields that could be used for signatures include the datagram size, identification, flags, fragmentation offset, protocol number, and the source and destination address. An example of an attack detectable by a signature at the IP level is the Land attack. The Land attack is a crafted packet that sets the source and destination IP address to the same value. The Land attack is able to lock up or reboot a variety of systems.

At the Transport layer the anomaly detection system is effectively the same as at the Internet layer, but the baseline statistics are generated from the Transport layer protocol headers, which are the payload of IP packets. Transport layer protocols include TCP, UDP, and ICMP [12]. A wide variety of combinations of header fields in the Transport layer could be used for signature detects including source and destination port, sequence numbers, acknowledgement number, data offset, or the flag bits. At this level an example of an attack that could be detected by signature is the well-known “Christmas Tree” attack. Its signature is that all of the bits in the flag field are turned on. As SANS notes, “There are

some flaky services that die when hit with these patterns [9]."

In the application layer, both anomaly and signature detection get a bit trickier because of the wide variety of possible protocols available. At this point both anomaly and signature based detection methods rely on "protocol analysis." The term "protocol analysis" means that the IDS sensor understands how various protocols work and can analyze the traffic of those protocols to look for suspicious or abnormal activity [10]. Also at this level, the data payload of the packet is available for analysis. This payload could be subjected to both of the analysis techniques based on keywords and language constructs. The SSH protocol, which is implemented at this level, by its nature limits the analysis that could be done because the payload and many of the SSH protocol headers are encrypted. Also lost with the encrypted data is the ability to record and replay a session in its entirety. Such ability is beneficial for evidence gathering, and can aid in the recovery of systems because it gives the computer security analyst a complete picture of the intruder's activities.

Proposed IDS with SSHD modified to be a sensor

As discussed previously, intrusions can be detected at all levels of the communication process except possibly at the application level where programs like SSH have features which make them impervious to protocol or keyword analysis. A sensor that was able to function at this level would supplement already proposed low-cost intrusion detection systems like the one presented by TJ Vanderpoel [10]. The following is a proposal of ways in which the SSH daemon could be modified to allow it to function as a sensor in an intrusion detection system, and a suggested architecture for its use. It is critical that any modification to the daemon be compatible with the protocol specification as defined by the SSH RFC.

The goal of this system is to preserve the security of the connection while allowing the computer security analyst (via the intrusion detection system) to perform signature, anomaly, or protocol analysis detection techniques to protect her system from misuse or intrusion.

Signature-based detection at this level of the protocol stack should include a keyword-based capability. The SSH daemon would be modified to decode the packet contents and pass the decoded clear text traffic though a keyword filter. If a suspicious string is detected in the packet, one of several possible reactions could be implemented depending on the administrator's needs. It could be possible to have a reactive system close the connection, or begin evidence gathering by recording the entire session. Keyword analysis filter is a flexible component as it could trigger on specific known compromise strings, or it could trigger on strings indicating a potential intrusion. For example, triggering on the string "~{" in relation to a tunneled ftp connection or seeing the string "mkdir …" would be known signatures. If the keyword appears in a potentially ambiguous

context, an alert could be sent to the analyst for further investigation. An example of this is the string “root” which could occur in a gardening discussion or some other innocuous exchange as well as in an attack scenario.

Anomaly detection could easily be preformed on the traffic available at this level. Instead of a statistical analysis of packet headers, user profiles of varying levels of detail could be generated. User profile elements could be derived from the meta-data of the connection, usage history, or stylistic elements of a users interaction. The meta-data of the connection would be elements such as the source of the connection, the destination beyond the initial system, time-of-day connected, and possibly the connection duration. Usage history could consist of a set of the user’s statistically “favorite” applications, which would note that a user uses vi as an editor instead of emacs, and has never used the editor pico. Finally, stylistic elements could be triggered by user habits. For example, predictably every time a given user logs in he accesses applications in the following order: mail, ls –laF, followed by a CAD application like the IDEAS software package. Deviations from the data gathered in the user’s profile would be analyzed for statistical significance, and, if a given threshold is exceeded, the activity could be reported as an anomaly that warrants investigation.

The functionality of this system is extensive given the ability to plug-in specific protocol analysis modules to detect signatures or anomalies in tunneled protocols. Potentially tunneled protocols include the CVS (Concurrent Versions System) protocol, X Windows, VNC (Virtual Network Computing), and FTP. This is not an exhaustive list, as many computing capabilities can be secured via the SSH protocols ability to tunnel arbitrary TCP/IP ports. An example of a detect at this level would be the triggering of an anomaly alert generated by the detection of a tunneled protocol not in a user’s profile.

The architecture that would make this intrusion detection system most effective is relatively straightforward. Access to the network should be limited to a few well-defined points in the perimeter. These entry points would consist of services running on so-called bastion hosts that provide externally required services. Bastion hosts are specially configured to the minimum requirements necessary to provide their service [13]. Services provided by bastion hosts include file transfer services, web services, and login services. The login service bastion hosts will run the modified SSH daemon so that all SSH based traffic coming into the network is visible to the sensors. The SSH daemon on the login hosts is configured to force password authentication. Forcing password-based authentication prevents a user from gaining access to the system with no challenge as could be the case if a cracker gets his hands on a user’s private key, and that private key was not protected by a passphrase. In addition to the security provided by limiting the services available on a bastion host, the login servers could make use of the “chrooted jail “ concept to provide extremely limited shell functionality. In this case, user activity will be limited to using SSH to login to an internal machine.

To fully incorporate this sensor into an enterprise intrusion detection system, it should have the capability to provide alerts to external analysis engines, databases, or intrusion detection consoles. A proposed Internet-Draft for an XML based format called IDMEF (Intrusion Detection Message Exchange Format) has been designed to enable this compatibility [14]. This format could be combined with an implementation of the proposed IDXP (Intrusion Detection Exchange Protocol) protocol, which is designed to transport alerts among elements of the intrusion detection system, such as sensors/analyzers, consoles, and managers [11]. A central management console, like ACID (Analysis Console for Intrusion Databases) could be used to collect IDMEF messages from the SSH daemon's embedded sensors [15].

It is important to note that the system described has protection elements built into it, and it should be able to be implemented with minimal impact on system performance. IDXP provides a secure mechanism of communication with data integrity, confidentiality, and mutual authentication mechanisms over a connection-oriented protocol. The only traffic generated by the system is the IDMEF alerts sent to the console, and possibly a copy of any traffic that is recorded for evidence, which would of course be moved in a secure fashion. The sensor code embedded in the SSH daemon would be optimized as much as possible to minimize its impact on the daemon's performance. If disk space and privacy concerns are an issue, traffic that does not trigger an alert should not be recorded.

Like many computer security tools, this system has the potential to be abused by personnel with access to the system. Administrators must take care when crafting a policy for its use. Like other intrusion detection systems, the Department of Energy's Network Intrusion Detector (NID) in particular, this system could be used to spy on the computer activities of employees or users. A complete picture of any user's activity can be gained especially if the ability to interpret tunneled protocols is added to the system. It would be possible to reconstruct entire X sessions giving the analyst, or whoever has access to the intrusion detection system, a complete view of the screen activities of the end user.

Conclusion

Applications which use clear text transmission of usernames and passwords should be a legacy of the past, since the SSH protocol provides effective, easy to use, secure remote access. Security, however, is a chain of elements and it is only as good as its weakest link. With remote access, the weakest link is the user. Therefore, the ability of SSH to subvert intrusion detection systems must be mitigated for some environments. Using the SSH daemon as a sensor in an intrusion detection system solves this problem by guaranteeing a secure connection, yet allowing signature, anomaly, and keyword detections through to

the highest level of the communication. This system is probably not appropriate for all sites, especially those where there are privacy or legal issues; however, it is appropriate for more security minded environments.

References

[1] “Sniffer.” Webopedia. Feb 5, 2002.
http://www.webopedia.com/TERM/s/sniffer.html (Mar 9, 2002)

[2] Sipes, Stephen. “Why your switched network isn't secure.” Intrusion detection FAQ. Sept 10, 2000.
http://www.sans.org/newlook/resources/IDFAQ/switched_network.htm (Mar 9, 2002)

[3] Delio, Michelle. “The Key to Encryption.” Wired News. Jun 22, 2001.
http://www.wired.com/news/ebiz/0,1272,44740,00.html (Mar 9, 2002)

[4] Brooks Suzukamo, Leslie. “Home Invasion.” Pioneer Press. Jan 28, 2002.
http://www.prod.twincities.com/mld/pioneerpress/business/technology/personal_technology/2555366.htm (Mar 12, 2002)

[5] Honeynet Project. “Know Your Enemy: Honeynets.” Dec 11, 2001.
http://www.linuxsecurity.org/feature_stories/feature_story-95.html (Mar 12, 2002)

[6] Cohen, Fred. “Introduction and Basic Idea.” Deception Toolkit
http://all.net/dtk/dtk.html (Mar 12, 2002)

[7] Ylonen, T. et al. “SSH Transport Layer Protocol.” SSH Internet-Draft. Feb 28, 2002.
http://search.ietf.org/internet-drafts/draft-ietf-secsh-transport-13.txt (Mar 12, 2002)

[8] Zirkle, Laurie. **“**What is host-based intrusion detection?” Intrusion Detection FAQ
http://www.sans.org/newlook/resources/IDFAQ/host_based.htm (Mar 12, 2002)

[9] SANS Institute. “Detects Analyzed 12/24/99.”
http://www.sans.org/y2k/122499.htm (Mar 12, 2002)

[10] Kent Frederic, Karen. “Network Intrusion Detection Signatures, Part 3” Feb 19, 2002
http://online.securityfocus.com/infocus/1544 (Mar 12, 2002)
[10] Vanderpoel, TJ. “Deploying Open Sourced Network Intrusion Detection for the Enterprise.” IntrusionDetectionFAQ. Mar 4, 2001
http://www.sans.org/newlook/resources/IDFAQ/open_source.htm (Mar 14, 2002)

[11] Feinstein, B, et al. “The Intrusion Detection Exchange Protocol (IDXP).” Jan

8, 2002.
http://www.ietf.org/internet-drafts/draft-ietf-idwg-beep-idxp-04.txt (Mar 14, 2002)

[12] "Transport Layer Protocols." TCP/IP For Internet Administrators.
http://www.pku.edu.cn/academic/research/computer-center/tc/html/TC0308.html
(Mar 15, 2002)

[13] Dillard, Kurt. "What is a bastion host." Intrusion Detection FAQ.
http://www.sans.org/newlook/resources/IDFAQ/bastion.htm (Mar 15, 2002)

[14] Curry, D. Debar, H. "Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition." Dec 28, 2001.
http://search.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-06.txt (Mar 15, 2002)

[15] Danyliw, Roman. "Analysis Console for Intrusion Databases."
http://www.andrew.cmu.edu/~rdanyliw/snort/snortacid.html
(Mar 15, 2002)

Last Updated: July 21st, 2017

Upcoming Training

SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
Community SANS Omaha SEC401*	Omaha, NE	Aug 14, 2017 - Aug 19, 2017	Community SANS
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
Community SANS Trenton SEC401	Trenton, NJ	Aug 21, 2017 - Aug 26, 2017	Community SANS
Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style	Virginia Beach, VA	Aug 21, 2017 - Aug 26, 2017	vLive
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 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 San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FL	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
Community SANS Albany SEC401	Albany, NY	Sep 11, 2017 - Sep 16, 2017	Community SANS
Mentor Session - SEC401	Ventura, CA	Sep 11, 2017 - Oct 12, 2017	Mentor
Community SANS Columbia SEC401	Columbia, MD	Sep 18, 2017 - Sep 23, 2017	Community SANS
Community SANS Dallas SEC401	Dallas, TX	Sep 18, 2017 - Sep 23, 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 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
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