IMPLEMENTING A TRUSTED THIRD-PARTY SYSTEM FOR SECURE SHELL

Sean Kavanagh

April 2017

ABSTRACT

From a 2003 paper titled "Security Implications of SSH" William Pfeifer wrote:

"There has been significant discussion on the whys and wherefores of the installation of SSH, the benefits of SSH over earlier protocols, and even how to set up tunneling to secure communication for other protocols via SSH. I have thus far seen little discussion on the potential security exposure offered by SSH and how to mitigate that exposure." [1]

At the time this was written, it was important for enterprises to not disregard the exposure a new technology introduced, even when that same technology improved upon the security of the status quo. Fourteen years later, Secure Shell (SSH) is the status quo. It's important today to not become complacent and to continue to investigate improvements in mitigating these known and sometimes permitted risks.

Although SSH is an improvement over previous protocols regarding encryption and integrity, it introduces exposures and makes administrative transparency more difficult. Accountability, non-repudiation, and data integrity are important elements that comprise a sound enterprise security model. These elements constitute a highly-flawed arrangement when dealing with remote administration. Data that can be audited and non-repudiated when transmitted over the network is at risk of being intercepted or compromised, while data that is integral and secured end-to-end is difficult to audit and can potentially be repudiated. This dilemma makes it a challenge to strike a balance between accountability and routine protections for secure communications. This paper will examine the risks associated with SSH and introduce concepts to mitigate them while facilitating workable administrative transparency through the development of a trusted third-party system.

The trusted third-party system will act as an intercepting proxy for SSH so that communication can be monitored and retained while still ensuring ample protection. Also, the system will add an additional layer of protection when combined with traditional secure network architectures.

HISTORY OF SSH

SSH was developed in 1995 by Tatu Ylönen a research student from Helsinki, Finland after his university became the victim of a credential-sniffing attack. He publicly released SSH-1 as freeware and it was quickly embraced at places like UC Berkley. Five years after SSH-1 was released its adoption grew to an estimated 2 million users worldwide. SSH became the standard way to remotely administer UNIX-based operating systems and Tatu Ylönen went on to found SSH Communications Security, a company that maintains a commercial version of SSH. He also was the main author of NIST IR 7966, which is the US Federal Government guidelines around SSH key management.[2] [3]

Today UNIX-based operating systems comprise 66.6% of the web servers on the internet and SSH is the prevalent means of their administration.[4] SSH may see more adoption in the future, as Microsoft announced bringing an SSH server to Windows.[5]

THE TRUSTED THIRD-PARTY SYSTEM

The trusted third-party system (TTP) is a web-enabled application that acts as an intermediary between administrative clients and the systems they access using SSH. The client sessions are proxied using a secure websocket connection over Transport Layer Security (TLS). The SSH connections themselves are established server-side to the protected servers. Since TLS can be decrypted at the application level, all communications can be secured and audited. Also, the application will control access to the systems themselves, ensure revocation of administrators, and centralize authentication. Figure 1 depicts an architecture in which the trusted third-party system is used as a proxy into systems hosted in a perimeter network. As a matter of convention, the author will refer to the trusted third-party system simply as "the control system."

Disclosure: The control system is based on work the author has implemented in an open source project named KeyBox.[6] The noted project is a starting point for such a system and is not complete with the features as described in this paper.

[1] Pfeifer, William (April 2003). "Security Implications of SSH" - http://www.sans.org/reading-room/whitepapers/vpns/security-implications-ssh-1180
[2] Ylönen, Tatu. "Tatu Ylönen, Inventor of SSH" - https://www.ssh.com/people/tatu-ylonen
[3] Barrett, Daniel J. & Silverman, Richard E. (February 2001). "SSH: Secure Shell – The Definitive Guide" - http://docstore.mik.ua/orelly/networking_2ndEd/ssh/ch01_05.htm
[4] W3Techs Web Technology Surveys (April 2017). "Usage of operating for websites" - https://w3techs.com/technologies/overview/operating_system/all
[5] Bright, Peter (June 2, 2015). "Microsoft bringing SSH to Windows and PowerShell" - https://arstechnica.com/information-technology/2015/06/microsoft-bringing-ssh-to-windows-and-powershell
[6] Kavanagh, Sean (2017). "KeyBox: Web-Based SSH Access and Key Management" - https://github.com/skavanagh/KeyBox

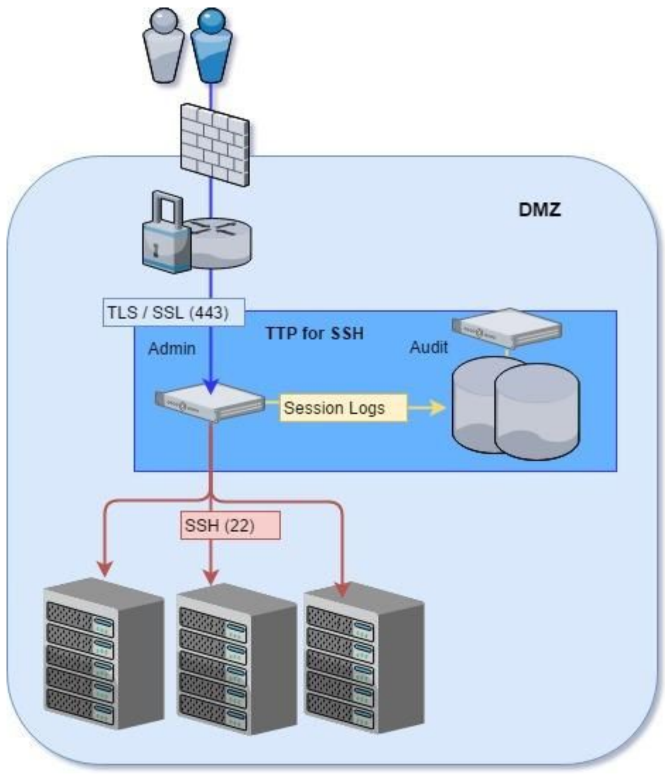

Figure 1 - Control System Architecture

PROTECTING SYSTEMS IN A PERIMETER NETWORK

Features of SSH can expose critical systems outside of protected network segments. Port forwarding through an SSH tunnel can allow for an administrative user to expose protected communication ports. Even the services listening on ports local to the system are at risk.

For example, if a Server Message Block (SMB) network share is protected so only systems in a perimeter network can access it, SSH can be used to bypass the protection through port forwarding.

```
ssh -L 445:[dmz-smb]:445 user@[dmz-server]
```
[7]

If the SSH session above is initiated outside the perimeter network, then SMB is also exposed to unauthorized systems outside the network segment.

Well protected networks segments are setup as Secure Enclaves and physically disable direct SSH access. Administration is done by first having the client establish a Virtual Private Network (VPN) connection to a server with means inside the enclave. This adds a layer of protection to systems hosted in the enclave so they cannot be exposed by SSH or other means. However, it does not address any auditing requirements. The control system could be added as an additional layer of protection to the secure enclave and addresses these auditing requirements. An Intrusion Detection System (IDS) could also be added to analyze all traffic passing through to the control system.

PROTECTING DATA IN A PERIMETER NETWORK

While SSH can prevent data from being intercepted by third-parties, there is a lack of indication when first-parties mishandle data. For example, a rogue user can very easily create a snapshot of a database and transfer the snapshot to an unsecured location through secure copy (SCP) or SSH file transfer (SFTP) protocols.

Also, the database itself can be exposed to clients outside a perimeter network through port forwarding.

[7] Kondratenko, Artem (March 2017). “A Red Teamer's guide to pivoting” - https://artkond.com/2017/03/23/pivoting-guide

```
ssh -L 1521:[dmz-db]:1521 user@[dmz-server]
```
[8]

If all administrative traffic is forced through the control system, databases cannot be exposed through port forwarding and data cannot be copied to unsecured locations using SSH without the control system allowing it.

AUTHENTICATION

The strongest type of authentication uses multiple factors. Public-key infrastructure (PKI) based x509 authentication involves a certificate authority (CA) to validate the user's public key. The corresponding private keys can be issued to smart cards or hardware tokens offering an additional factor of authentication. Some SSH implementations do not support x509 based authentication outright.[9] Even when x509 is supported, this authentication must be configured at the system level, which means the system can be configured to still allow authentication by weaker means. These weaker authentication methods can include plain passwords, SSH keys, or even backdoors into the system.

The value in obtaining SSH keys has gained the attention of malicious hackers and even nation states. In 2014, a malware known as "The Mask" included the collection and exposure of private SSH keys. This malware was in operation for at least 7 years before it was discovered and is largely presumed to be the work of a nation state.[10] Also, in an attack that was believed to be sponsored by North Korea on Sony Pictures, private SSH keys were included in the list of comprised files.[11]

The control system would be a centralized way for administrative users to authenticate. Ideally this system would be outside of the administrator's control. Authentication could easily be integrated with directory servers or include multi-factor authentication such as public-key infrastructure (PKI) with smart cards. Most importantly this authentication ensures that all private keys used for system access are secure. This centralized authentication is enforced when layering the control system behind a perimeter network that physically disables SSH. Administrative sessions can always be tied back to the actual users through the control system, even when using shared accounts on the target systems themselves. Additionally, intrusion detection is made simpler, since things like failed login attempts can be monitored at a single location.

PRIVILEGE ESCALATION

Since it's difficult to monitor SSH sessions, there is an increased risk of privilege escalation on the systems themselves. Once a user obtains the necessary privileges, they have full control of the system and can typically hide any trace of their activity. While privilege escalation is not a direct result of exploiting SSH itself, SSH acts as a cover for these activities and should highlight the importance of monitoring remote administration.

Any program or script that is run as the privileged user and also writable to a non-privileged user is at risk of privilege escalation. Commands can be overridden and used as back doors for malicious users to gain access to privileged accounts.

For example, the following could be inserted into a script that is run by the privileged user.

```
if [ $EUID -e 0 ];  then
        cat /tmp/Sd5ad34sE/master_key.pub >> /root/.ssh/authorized_keys;
done;
```

This snippet checks to see if the script is currently executed as the root user and adds a public SSH key to gain access as that user. This can also be done with programs executed from a scheduled job running as the privileged user.

Not only are permissions to executables vulnerable, but paths are as well. Any directory on the privileged user's path that is writable to other users could possibly contain executables that override other commands and escalate privileges.

Programs used to limit privileged access can also be used to obtain privileged access, as in the case with sudo (superuser do). The sudo utility allows commands and programs to be executed with permissions of a more privileged user. These privileges are typically maintained in a configure file (/etc/sudoers) on the system itself. While sudo has some security benefits by encouraging users to limit privileged access, it tends to be difficult when enforcing only the essential privileges the user requires. Furthermore, it moves the assignment of privileges outside of the enterprise's purview. This lack of central control means that

---

[8] Deng, Zemian (November 2014). "How to use SSH tunneling to get to your restricted servers" - https://www.javacodegeeks.com/2014/11/how-to-use-ssh-tunneling-to-get-to-your-restricted-servers.html
[9] Morgan, Iain (June 2010). Public mailing list - http://lists.mindrot.org/pipermail/openssh-unix-dev/2010-June/028702.html
[10] Kaspersky Lab (February 2014). "Unveiling Careto – The Masked APT" http://kasperskycontenthub.com/wp-content/uploads/sites/43/vlpdfs/unveilingthemask_v1.0.pdf
[11] /r/Hacking (2015). Reddit - https://www.reddit.com/r/hacking/comments/2n9zhv/i_used_to_work_for_sony_pictures_my_friend_still

there must be complete trust with the user administrating the system or systems to maintain proper configuration.

Lastly, programs that seem harmless can be used as backdoors into the privileged user's shell using sudo.

```
sudo find <filename> -exec bash \;
sudo vim [ESC KEY], [TYPE] :!bash
sudo less <filename>, [TYPE] !bash
```

This makes enforcement of privileges cumbersome. Not only does one have to ensure that only the necessary programs are set for a user, they also have to ensure those programs do not allow for privileged access to other shells or programs.

These examples are just a few trivial ways that privileged access can be obtained. Once privileged access is obtained, the user can manipulate footprints or history retained on the system. Again, this isn't a direct result of SSH, but should empathize the importance of setting up a control system to monitor these activities.

AUDITING

Since SSH is encrypted end-to-end from the client to the server, centralized auditing becomes difficult. Typically audit logs, audit settings, and privileges to those logs and settings are maintained on the systems themselves. Audit logs tend to lack details, are at risk of being incomplete, and are susceptible to manipulation. Also, audit logs will not have a complete picture of what has been done across the enterprise. For example, if an enterprise wanted to look into the failed login attempts during a set period of time across all the systems, they would have to pull the /var/log/auth.log on every system to compile a list.

Since the control system acts as an intermediary for SSH, it could centrally collect and retain what is being done on all systems. Data from an administrative session can be reviewed at will. Sessions that touch on known files or commands that are susceptible to manipulation could be flagged for review. For example, editing /etc/password, /etc/group, or adding a public key to authorized keys could trigger a review of the session. A future enhancement could use the culmination of all data to establish a normal pattern of administration for the enterprise. As the control system collects auditing data it could flag administrative sessions for additional review if that session falls outside of the established pattern norms.

SUMMARY

There is little systematic protection when dealing with SSH and most likely any defense that enhances transparency would have to be carried out procedurally. An example of such, is the "two-person rule", which was implemented at the National Security Agency (NSA) to prevent privilege escalation allowing for unauthorized access to classified information.[12] It must be recognized as a limitation when systems can only be secured procedurally and this limitation must be recognized with systems administered through SSH. Implementing a trusted-third party system that layers TLS on top of SSH allows for immediate systematic protection and new capabilities around administrative auditing which provides much needed transparency.

[12] Greenberg, Andy (June 2013). "NSA Implementing 'Two-Person' Rule To Stop The Next Edward Snowden" - https://www.forbes.com/sites/andygreenberg/2013/06/18/nsa-director-says-agency-implementing-two-person-rule-to-stop-the-next-edward-snowden