How and Why More Secure Technologies Succeed in Legacy Markets: Lessons from the Success of SSH

How and Why More Secure Technologies Succeed in Legacy Markets: Lessons from the Success of SSH

Nicholas Rosasco
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
nrosasco@acm.org

David Larochelle
Department of Computer Science
University of Virginia
larochelle@cs.virginia.edu

Abstract

Secure shell (SSH) can safely be called one of the rare successes in which a more secure technology has largely replaced a less secure but entrenched tool: telnet. We perform a market analysis to determine how and why SSH succeeded despite the existence of an entrenched legacy tool while similar technologies such as secure file transfer protocols have been far less successful. We show that network externalities, usually a first order effect, were not a significant factor impeding the adoption of SSH, and that SSH offered equivalent functionality and greater ease of use. We argue that these factors were the primary consideration in the willingness to change. Additionally, we argue that the openness of the standard, which facilitated the creation of numerous compatible implementations, was a key element in the economic decision made by system administrators.

Introduction

Secure shell (SSH) can safely be called one of the rare successes in which a more secure technology has largely replaced a less secure but entrenched tool: telnet. Since the early commercial and later open source versions in the mid ‘90s, the tool, created as a replacement for telnet and the rsh/rlogin/rcp trio, has become the method of choice for remote login and X tunneling and is a rapidly becoming one of the most pervasive applications for encryption technology outside of embedded systems, particularly after being freed from RSA related patent complications [Bertrand99].

It is a non-trivial task for an installed base to be brought to a new client, no matter how similar it is. The size of a system’s login user base with a need for interactive OS access can be large, especially for large ISPs and development projects. Even with the necessary technically savvy; inertia and a preference for known tools must be overcome no matter

how beneficial the software change is perceived to be. We perform a market analysis to determine how and why SSH succeeded despite the existence of an entrenched legacy tool while secure file transfer technologies have failed to displace FTP.

Background

SSH was developed by Tatu Ylönen after his university was the victim of a password sniffing attack [Barrett01, p. 9]. SSH was first released in July of 1995. The initial version was designed as a drop in replacement for the rcp/rsh/rlogin trio, and offered similar interface functionality to the current version, including X tunneling [Ylönen95]. Many in the security community responded positively to the release, and by the end of the year around 20,000 users were using SSH. Ylönen was also receiving around 150 email messages a day requesting support. He founded SSH Communications, Ltd. to commercialize the software [Miller02] and handle the flow of email [Barrett01, p. 10]. After SSH Communications began to release versions of SSH under increasingly restrictive licenses, OpenSSH, an open source version based on a liberally licensed earlier release of SSH, was created. This, combined with the expiration of the RSA algorithm patent and the relaxation of export rules regarding cryptography by the United States, has allowed OpenSSH to be freely distributed on the distribution media for many Unix distributions [Bertrand99]. The growth of SSH has continued. It was estimated that by the end of 2000 there were 2,000,000 users of SSH [Barrett01, p. 12]. Though OpenSSH is now the most widely used SSH server, the SSH Communications product still has significant market share [OpenBSD03].

The creation of the OpenSSH project has no doubt contributed to the popularity of SSH. But it was SSH's early popularity in the security community that prompted the development of OpenSSH. Theo de Raadt, the founder of both OpenBSD and OpenSSH, claimed that for the two years before the release of OpenSSH in 1999, the first thing many users did after installing OpenBSD was to install SSH [Bertrand99].

While SSH servers primarily run on Unix systems, many users need to connect to these servers from non-Unix systems. It is worth noting that SSH succeeded as a cross platform tool for interactive login -- numerous clients are available for many operating systems -- without the help of Microsoft. Microsoft continues to distribute a telnet client with Windows but does not distribute an SSH client.[1]

Analysis

We can view technical staff and their management as rational actors with an interest in the efficient achievement of security for the organization. Removing telnet entirely can be seen as promoting a socially optimal outcome by preventing careless users from placing others at risk. In their rationality, we see them balancing the needs of functionality and ease of use with their desire to achieve security.

---

[1] We were told that Microsoft decided not to include SSH in Windows XP because of legal complications involved with cryptography and because there were many freely available SSH clients for Windows [Moore01].

Network externalities are often a factor in technology driven products [Shapiro99, Katz94]. Metcalf's law states that the benefit each user gets from a network is proportional to the square of the number of users [Shapiro99, p. 184]. This makes it difficult for new technologies to get established but can result in exponential growth curves for technologies that are becoming established. At first glance, network effects appear to be a significant impediment to widespread adoption of SSH. SSH clients are only useful if there are SSH servers, and servers are only useful if there are clients – the traditional "chicken and egg" problem. However, most users only have shell accounts on a few systems – and many have only one shell account. The utility a user derives from using an SSH client is determined by whether the few systems they access have SSH servers, not by the total number of SSH servers. Positive feedback still exists to an extent. For example, as expertise in SSH becomes more wide spread, this adds value to SSH. Additionally, after SSH reached the point where installing it was considered a standard best practice, its adoption increased further.

The cost of a security failure is borne by both the users and the owner/administrator of a system. On multi-user UNIX systems, once an attacker is able to obtain user-level access, it is often possible to obtain root privileges relatively easily – witness, as a classic example, the Emacs exploitation made famous by Stoll in *The Cuckoo's Egg* [Stoll89]. It is therefore entirely possible that system administrators and other users on the system will suffer because of a single sniffed password. Even in the absence of extensive system damage or theft of information, they are deeply inconvenienced by the ensuing efforts necessary to restore faith in the system. At the corporate or institutional level, the entire organization suffers – particularly since the compromised system could have become a beachhead for further penetrations. System administrators and their managers had an incentive to install and promote SSH, but faced the challenges not just of software replacement, but also user training [Hatch02]. However, a desire to increase accountability may have provided a further incentive to use SSH. In the event that a user account was compromised, SSH made it possible to push more culpability to the user for her password behavior. Without SSH, it was difficult to determine if the user was careless with her password, or if it was sniffed over the network.

Since SSH required both a client and a server, the existence of multiple clients, especially open source clients, helped limit the risks of a switch to SSH. To some extent, the existence of multiple clients was facilitated by the open nature of the protocol (a simple search reveals more than a dozen clients available from a wide range of sources [Google03]), which permitted multiple versions of the server application as well. Through the eventual absence of patent or other intellectual property protection what began as a commercial product became an equally no-license-cost option [Bertrand99][2].

While password sniffing was not necessarily a common occurrence, and the processor capacity necessary to support the encryption not always trivial, the security advantages of SSH over telnet could be specified precisely. This avoided the customary imperfect information problem, which remains significant in information security. The software

[2]telnet was also unencumbered intellectual property restrictions.

development community – particularly developers of open solutions, but also proprietary developers – should view this type of easy-to-compare situation as an invitation to an opportunity to compete. If a competitor has a solution that is known to be insecure, developers should aim to create a drop-in alternative. This is particularly the case if the developer can achieve a point of minimal added cost, such as most of the current popular incarnations of SSH over the equally free telnet solutions.

Aside from the security issues involved in the decision to install SSH, functionality was the same though communication and system performance were impacted by the overhead of encryption. The principal life cycle cost was installing SSH and replacing telnet-only clients at the user's end. This was more significant when cryptographic export and intellectual property issues made installation more of a hassle. For users of X Windows, functionality was enhanced: SSH could automatically handle the display of remote programs on a local desktop. The existing protocols required the user to manually adjust the settings each session. This feature of SSH constituted a major convenience and usability improvement, making SSH a powerful tool for remote applications. This is another important point for the author of a potential competitor to an insecure legacy program, and an equally significant point for an advocate of creating security conscious consumers. "Secure" can, and perhaps should, be easily advocated when it comes with greater functionality.[3]

Secure File Transfer

The several technologies for secure FTP replacement are "also-ran" solutions that never achieved the same level of acceptance as SSH. Secure file transfer solutions are implemented by a wide selection of server and client applications but lack a single distinct standard. The various secure file transfer options include two semi-distinct protocols supported by tools within the OpenSSH and commercial SSH tool suites: secure copy (scp), a relative of the remote copy system, and sftp, which is loosely based on the tradition FTP protocol[4]. Both of these protocols rely on SSH[5] to provide an encrypted stream [Barrett01].[6] scp was designed to be a drop-in replacement for rcp and was included with the UNIX version of SSH[7]. It has largely supplanted rcp. However, scp does not provide the power and ease of use of ftp. With early versions of scp there were

---

[3] See [Larochelle03] for a discussion of the interaction between security and functionality.

[4] These protocol standards are documented at:
http://www.ietf.org/proceedings/01aug/I-D/draft-ietf-secsh-filexfer-00.txt and
http://www.openSSH.com/txt/draft-ietf-secsh-filexfer-02.txt.

[5] SSH provides the ability to establish secure connections between arbitrary ports on different systems. However, the idiosyncrasies of FTP prevent this feature from being used to effectively secure the protocol. ([Barrrett01] includes a lengthy section describing methods to establish a fully encrypted FTP session using SSH. But their methods will not work on all systems. The authors acknowledge that because of the complexity, their solution is more of a "parlor trick [to use] at geek parties" than a practical means of obtaining secure file transfer.)

[6] Technically, these protocols will provide secure file transfer over any encrypted stream, but we are not aware of any implementation that does not use SSH for encryption and authentication.

[7] The version of scp included with SSH2 is actually implemented using sftp internally; however, this version still presents the same rcp-like interface as the version in SSH1.

widespread compatibility problems (significantly more so than those between the implementations of SSH1 and SSH2 protocols). sftp, which is included with SSH2, provides an ftp-like interface for secure file transfer. In addition to these two standards, there is also competition from products offering similar file transfer functionality, including various commercial entities that create "secure FTP" products that include security extensions to the existing FTP protocol and are often not compatible with either scp or sftp [Barrett01, p. 379]. These solutions represented the best efforts of several companies and groups to provide secure authentication and transmission of files from one system to another. The failure of the practitioners, system administrators and their employers, to adopt these as alternatives to the widely used clear-text FTP protocol provides a striking contrast to the near universal acceptance of SSH.

We see the inclusion of secure file transfer options into SSH as both facilitating and inhibiting the spread of secure file transfer programs. These tools are now available on many Unix systems. However, these protocols have not been widely implemented by other SSH clients. The relatively small number of graphical clients supporting these protocols is particularly problematic. Ironically, the success of early versions of SSH may have actually hurt the adoption of secure alternatives to FTP by providing some security functionality to technically savvy security conscious users through scp, which is an adequate replacement for rcp but lacks the functionality of FTP. The fact that an SSH server is required to run these tools may have also limited their appeal and prevented their emergence as first-class Internet services. Additionally, while SSH was intended to supplant the rlogin/rcp/rsh trio, these tools cannot be seen as true drop in replacements since they are not backwards compatible with FTP.

Key to the deafening silence facing scp and sftp, as opposed to the chorus of approval for secure interactive logins, is the lack of a "killer" functionality such as simplified X tunneling offered by SSH. For many systems, the perceived need for secure file transfer was significantly less than the perceived need for secure interactive login. Many users had FTP accounts on systems on which they did not also have shell accounts.[8] Unlike a sniffed telnet password, the damage from a compromised FTP password was often largely limited to the individual user. Additionally higher cost versus functionality (particularly the transfer speed impact of adding encryption), no doubt also hindered the selection of any of the secure alternatives to the sniffable FTP protocol. Without a compelling selling point, managers and system maintainers alike were loathe to take a lengthy and unpleasant client and server application plunge, especially when given the blurry availability and industry picture evident in the early days. While the confusion was certainly not unique to secure file transfer applications, the additional murkiness of the end-user choices also clouded the chance of acceptance by users and system operators alike.

Early adoption of the secure file transfer options was also complicated by insufficient "open" availability, consumer confusion over availability, and commercial versus open implementation questions. The welter of difficult to distinguish options and their relatively recent entry into the Internet protocol environment no doubt also hindered the displacement of clear-text protocols with encrypted communications. The absence of a

[8] It is difficult to configure SSH to allow a user sftp access but not shell access [Barrett00].

broad selection of clients, particularly those geared for general users, for the early releases of the secure transfer protocols no doubt also made adoption more sluggish.

It is easy to think what might have been (the gradual supplanting of a clear text password network application), but also easy to spot where the various secure file transfer options did and did not offer the advantages of SSH: no increased functionality, even higher cost (far fewer client options) to switch, and a serious set of difficulties over the standard and complications with non-cooperative versions. However, we remain optimistic about the future of secure file transfer. With the growing acceptance of a standard and easier management implicit in the growing adoption of the OpenBSD team's implementations of the SSH2 suite, secure file transfer may achieve broader acceptance. The advent of a full-fledged secure FTP-like program (versus the less familiar rcp and scp) is relatively recent. The new application has yet to have a chance to achieve the same acceptance as the supplanter of telnet. Though we expect that many systems will continue to run FTP into the foreseeable future even with complementary secure alternatives, the spread of this new application is likely to be a bellwether for the acceptance of secure alternatives, and serve as a focal point in the overly fragmented area of secure file transfer protocols.

Conclusion

SSH provided superior security while current functionality was maintained. SSH's acceptance is demonstrated by the fact that installing SSH as an alternative to telnet is now widely considered to be a minimal security practice. The removal of telnet clients is now seen as a best practice [Fenzi02], and this view has further increased the adoption of SSH.

Similar technologies such as secure file transfer protocols provide similar benefits but have not achieved nearly the same level of acceptance as SSH. We have performed an economic analysis to determine why telnet has been largely supplanted by SSH but FTP remains widely used. The consequences of a security breach exploiting clear text passwords is far reaching -- the entire system is placed at risk. In many cases the risks posed by telnet and FTP were the same, but the perception of the costs to change obviously differs. An organization that provides shell accounts is likely to have an interest in the integrity of user data that extends beyond concerns for reputation and liability alone, and is responsible for the difficulties and costs of the switch. How and more importantly why, does the market view one security solution as achievable, and yet ignore the other? We have attempted to find lessons to be learned about the tradeoffs that are made, and how the secure option can be made more attractive. We have shown that network externalities, usually a first order effect, were not a significant factor impeding the adoption of SSH, and that SSH offered equivalent functionality and greater ease of use. These factors were the primary consideration in the willingness to change. Additionally, we believe the openness of the standard, which facilitated the creation of numerous compatible implementations, was a key element in the economic decision made by system administrators.

Acknowledgments

We would like to thank Joel Winstead, Nicolas Christin, Anthony Wood, Shiva Prasad and the anonymous reviewers for their helpful and insightful comments.

Bibliography

[Barrett01] Daniel Barrett and Richard E. Silverman, SSH, the Secure Shell: The Definitive Guide, USA: O'Reilly & Associates, (2001).

[Barrett00] Daniel Barrett and Richard E. Silverman, SSH Frequently Asked Questions, (Oct. 2000), http://www.snailbook.com/faq/restricted-scp.auto.html

[Bertrand99] Louis Bertrand, "How SSH was freed", Daemon News (Dec. 1999), http://www.daemonnews.org/199912/openSSH.html.

[Fenzi02] Kevin Fenzi and Dave Wreski. "Linux Security HOWTO", (June 2002), http://www.tldp.org/HOWTO/Security-HOWTO/index.html

[Google03] Google, Inc. "Google Directory: SSH Clients", (2003), http://directory.google.com/Top/Computers/Security/Products_and_Tools/Cryptography/SSH/Clients/

[Hatch02] Brian Hatch, "Greasing the Squeaky Wheels", ITWorld.com, (September 2002), http://www.itworld.com/nl/lnx_sec/09172002/.

[Katz94]Michael L. Katz and Carl Shapiro, "Systems Competition and Network Effects", The Journal of Economic Perspectives, Vol 8, (Spring 1994).

[Larochelle03] David Larochelle and Nicholas Rosasco, Towards a Model of the Costs of Security, (May 2003), http://www.cs.virginia.edu/larochelle/securitycosts.

[Miller02] Damien Miller, SSH tips, tricks, and protocol tutorial, (August 2002), http://www.mindrot.org/~djm/auug2002/ssh-tutorial.pdf

[Moore01] Jason Moore, personal communication, (February 2001).

[OpenBSD03] OpenBSD, SSH usage profiling, http://www.openssh.org/usage/, statistics listed as of May 2003.

[Shapiro99] Carl Shapiro and Hal R. Varian, Information Rules: a Strategic Guide to the Network Economy, Harvard Business School Press, (1999).

[Stoll89] : Clifford Stoll, The Cuckoo's Egg: Tracking a Spy through the Maze of Computer Espionage, New York : Doubleday, (1989).

[Ylönen95] Tatu Ylönen, Usenet posting of the SSH release announcement, (July 1995), message archived at http://groups.google.com/groups?hl=en&lr=&ie=UTF-&selm=YLO.95Jul12234021%40shadows.cs.hut.fi.