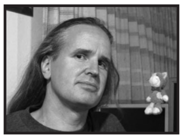
Do Users Verify SSH Keys?

PETER GUTMANN



Peter Gutmann is a researcher in the Department of Computer Science at the University of Auckland. He works on design and analysis of cryptographic security architectures and security usability, which includes lots of grumbling about the lack of consideration of human factors in designing security systems.
pgut001@cs.auckland.ac.nz

Abstract

No.

Discussion

Anecdotal evidence suggests that the majority of users will accept SSH server keys without checking them. Although SSH users are in general more security-aware than the typical Web user, the SSH key verification mechanism requires that users stop whatever they're trying to do when connecting and verify from memory a long string of hex digits (the key fingerprint) displayed by the client software. A relatively straightforward attack, for the exceptional occasion where the user is actually verifying the fingerprint, is to generate random keys until one of them has a fingerprint whose first few hex digits are close enough to the real thing to pass muster [1].

There are even automated attack tools around that enable this subversion of the fingerprint mechanism. The simplest attack, provided by a man-in-the-middle (MITM) tool called ssharpd [2], uses ARP redirection to grab an SSH connect attempt and then reports a different protocol version to the one that's actually in use (it can get the protocol version from the information passed in the SSH handshake). Since SSHv1 and SSHv2 keys have different fingerprints, the victim doesn't get the more serious key-changed warning but merely the relatively benign new-key warning. Since many users never check key fingerprints but simply assume that everything should be OK on the first connect, the attack succeeds and the ssharp MITM has access to the session contents [3]. (Since ssharp is based on a modified, rather old, version of OpenSSH, it'd be amusing to use one of the assorted OpenSSH security holes to attack the MITM while the MITM is attacking you.)

```
> ssh test@testbox
The authenticity of host 'testbox (192.168.1.38)' can't be established.
RSA key fingerprint is 86:9c:cc:c7:59:e3:4d:0d:6f:58:3e:af:f6:fa:db:d7.
Are you sure you want to continue connecting (yes/no)?

> ssh test@testbox
The authenticity of host 'testbox (192.168.1.38)' can't be established.
RSA key fingerprint is 86:9c:cc:d7:39:53:e2:07:df:3a:c6:2f:fa:ba:dd:d7.
Are you sure you want to continue connecting (yes/no)?
```

Figure 1: Real (top) and spoofed (bottom) SSH servers

A much more interesting attack can be performed using Konrad Rieck's concept of fuzzy fingerprints. Fuzzy fingerprints are SSH key fingerprints that are close enough to the real thing to pass muster, and as with the standard SSH MITM attack there's a tool available to automate the process for you [4]. This attack, illustrated in Figure 1, takes a target SSH server key and generates a new key for which the fingerprint is close enough to fool all but a detailed, byte-for-byte comparison. (Because the key doesn't have to be secure, merely to work for the RSA computation, you can simplify the key generation to require little more than an addition operation. The rate-limiting step then becomes the speed at which you can perform the hashing operation, and even there you can pre-compute almost everything but the last hash block before you start, making the key-search process extremely quick.) Since few users are likely to remember and check the full 40-hex-digit fingerprint for each server they connect to, this, combined with ssharpd, is capable of defeating virtually any SSH setup [5].

When the SSH fuzzy fingerprint work was first published, I wanted to run a real-world evaluation of its effectiveness, so I approached two large organizations with several thousand computer-literate (in some cases highly so) users to see if I could use their servers for the test. In order to determine the base rate for the experiment, I asked the IT support staff how many users had called or emailed to verify the SSH server key whenever it had changed in the past several years. They were unable to recall a single case, or locate any records, of any user ever verifying any SSH server key out-of-band. As the base rate for verification of completely different key fingerprints was already zero there didn't seem to be much to be gained by running an experiment with approximately matching fingerprints, since the result couldn't be worse than zero.

Conclusion

This result represents good news for both the SSL/TLS PKI camps and the SSH non-PKI camps, since SSH advocates can rejoice over the fact that the expensive PKI-based approach is no better than the SSH one, while PKI advocates can rest assured that their solution is no less secure than the SSH one.

References

[1] Dan Kaminsky, "Black Ops 2006: Pattern Recognition," 20th Large Installation System Administration Conference (LISA '06), December 2006: http://usenix.org/events/lisa06/tech/#friday.

[2] Sebastian Krahmer, "SSH for Fun and Profit," July 1, 2002: ftp://ftp.pastoutafait.org/pdf/ssharp.pdf.

[3] Jon Erickson, *Hacking: The Art of Exploitation*, No Starch Press, 2003.

[4] "THC Fuzzy Fingerprint," October 25, 2003: http://www.thc.org/thc-ffp/.

[5] Plasmoid, "Fuzzy Fingerprints: Attacking Vulnerabilities in the Human Brain," October 25, 2003: http://www.thc.org/papers/ffp.html.