Signed Archives: An evaluation of Internet trust

Jose Nazario
jose@monkey.org

April 14, 2003

1 Abstract

In 2002, a series of high profile compromises of Internet software servers resulted in the alteration of software archives. This prompted an evaluation of the state of trust of the signed software distribution system. Over 2800 archives representing over 1400 unique software packages were downloaded and their corresponding signatures evaluated for validity. These software packages were pulled from over 260 different sites and the keys retrieved only during the verification stage. Of the over 2800 archives checked, only 5 errors were found, three of which were found to be false negatives. Additionally, the characteristics of the keys used to sign these archives along with the key distribution systems were studied. These findings highlight weaknesses in the signed archive distribution system and demonstrate clear vulnerabilities facing several projects.

2 Introduction

In mid 2002, a series of compromises of high profile software distribution sites occured. This list includes the breakin and modification of the popular IRC client 'irssi' [1], the dsniff, fragroute, and fragrouter source code from Dug Song [2], and the OpenSSH source code [3]. In each case the modifications were detected via cryptographic checksums, but could have been detected with public key signatures for the packages, as well.

The use of public key signatures for software archives is popular, typically using the PGP standard. The model used in PGP (and tools using this standard, including GnuPG and openPGP) use the 'web of trust' model (see [4]). In this scenario, the author or team of authors generate a public and private key pair and sign the software using their private key. The public key is distributed and often signed by others as to its veracity and trustworthiness. When the software is downloaded, the public key is used to validate the source of the software and the match between the signature and the archive. This public key has increased trust based upon the number and nature of the signatures it has. More signatures obviously help raise the ease of verification of the key, and more signatures that are from well known keys raises this metric even further.

The modification of these popular software packages raises the question of how many other software packages are compromised but have remained unreported. Additional questions raised include the integrity of the underlying public key system, including the strength of the keys, the trustworthiness of them through signatures on the keys, and the key distribution mechanisms typically found.

In order to obtain the answers to these questions, a diverse set of signed software packages were identified and downloaded along with their signatures. The archives were then verified using the indicated keys and the result was recorded. While only a handful of negative results were found, weaknesses in the system overall were identified and bulk public key statistics were measured. Based upon this evidence, the public key signature system as it is used contains minor weaknesses and is susceptible to manipulation by an attacker.

3 Survey Results

In mid august, 2002, a total of 2804 signed software archives were identified via a google search (see *Methods*). This number represented 1426 unique archives, with the difference being due to software mirrors of some packages. These archives were located on 166 unique servers throughout the world. The software was downloaded using a cable modem and took approximately 2.5 days, with the archive and their signatures

using approximately 9.5 GB of disk space. During the bulk processing 2799 downloaded archives were a positive verification.

Five failures were found in the course of this batch verification. Of those, one was due to a truncated download (identified by comparing the archive against the same archive from different sites) and two were false negatives revealed by repeating the verification process. The remaining two failures were legitimate mismatches between the signature and the archive, which were in subsequent releases of the same software package. The author was contacted and this result was confirmed.

During the course of this study some archives were unable to verify from the **cmu-snmp** archives. These are found at `ftp://ftp.andrew.cmu.edu/pub/snmp/`. Attempts to verify the signature using the downloaded archive produce the error:

```
gpg: Warning: using insecure memory!
gpg: please see http://www.gnupg.org/faq.html for more information
gpg: Signature made Mon Mar 31 18:18:58 1997 EST using RSA key ID 65965CD1
gpg: Can't check signature: public key not found
```

An attempt was made to fetch this key from the various keyservers, all of which yielded errors:

```
$  gpg --keyserver PGP.mit.edu --recv-key 65965CD1
gpg: Warning: using insecure memory!
gpg: please see http://www.gnupg.org/faq.html for more information
gpg: requesting key 65965CD1 from HKP keyserver PGP.mit.edu
gpg: key 65965CD1: no valid user IDs
gpg: this may be caused by a missing self-signature
gpg: Total number processed: 1
gpg:           w/o user IDs: 1
```

This leaves the key unimported into the local key ring, causing the first error shown above. Examination of several archives and websites were unable to produce a key. All other archives were able to be processed by the bulk verification tools used in this study.

4 Uncovered Weaknesses

While at first it may appear that the trust in the system of signed archives on the internet is well placed, several common practices were revealed during this study. These observed behaviors can be combined to weaken the trust which can be placed in the signed archive. These practices include the placement of the public key, the vulnerability of the key to compromise, and a low number of signatures on an average public key. Each of these concerns is discussed below using data gathered from the public keys used in this study.

4.1 Inline Key Distribution

Dhe location for the distribution of the public keys associated with a signed archive is an important consideration. As described above for the **cmu-snmp** archives, public keys must be readily identifiable and downloaded in order to verify the signature. This concern was described by Alex Brennen in the 'Strong Distribution HOWTO' [5] in section 2.2:

> There are three steps that I recommend that you take in order to circulate your public key. First, you should post your public key on the website where the software is distributed from. You should place the ASCII armored public key is a conspicuous place where people can easily find and download it.

Care must be taken, however, to protect the integrity of the key. This is somewhat addressed in the 'Strong Distribution HOWTO' when the author states later in that same section,

> I do not recommend that you include your public key inside your software archive. While there is no technical security problems with this, it does encourage the end user to accept the public key driven by trust based in the location of the key rather than integrity imparted upon the key by signatures. Encouraging such habits in the end users will make them more susceptible to trojan horse attacks against the Strong Distribution Model in which fake archives and fake keys are distributed.

While Brennen cautions against inclusion of the key within the archive, distributing the key inline with the archive by placing it on the same server and often in the same directory is a similar action. A compromised server can not only have its software archives modified by an attacker but the keys forged, as well, leading to a match between the signature, key, and archive. Instead, the distribution site for the keys should be an additional factor into their trust. A key server which protects the keys protects the signatures, by extension.

Several popular software packages which are signed use inline key distribution, in addition to a key server, to make their public keys available. These products include:

- OpenSSH portable, which includes a copy of the maintainer's key as `OpenSSH/portable/DJM-GPG-KEY.asc`
- SSH Communications' SSH source code packages, which distribute several keys alongside their source code packages as `pub/ssh/SSH-DISTRIBUTION-KEY.asc`, `pub/ssh/SSH1-DISTRIBUTION-KEY-RSA.asc`, `pub/ssh/SSH2-DISTRIBUTION-KEY-DSA.asc`, and `pub/ssh/SSH2-DISTRIBUTION-KEY-RSA.asc`.

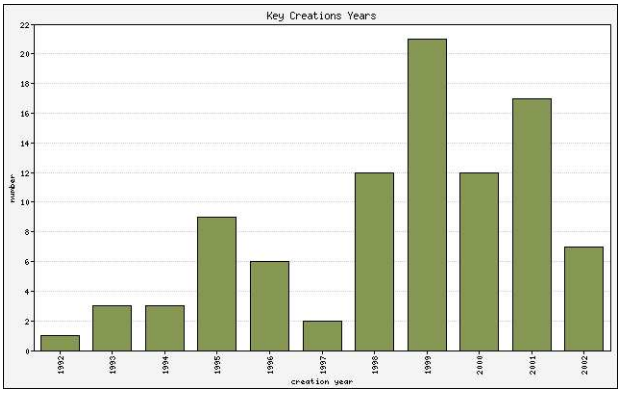

Figure 1: Distribution of key ages.
The keys used in this study were examined to find their creation year. This date was then graphed as a function of the frequency of the date of key creation. Most keys are no more than 3 years old, with some keys almost 10 years old.

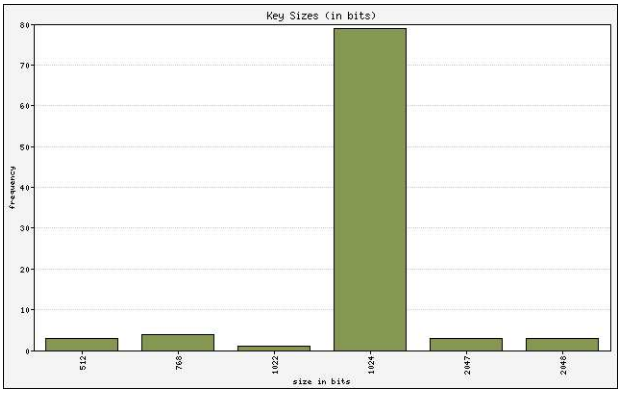

Figure 2: Size distribution of keys in use.
The sizes of the public keys used to sign the archives studied in this research are shown above. Of the 93 keys examined here, 79 have a size of 1024 bits. This graph does not differentiate between DSA keys (57 in this study) and RSA keys (36 in this study).

- the Cyrus mail tools packages, which distributes the keys used to sign their software in a file `KEYS`
- the Gnuplot project, which distributes the keys it signs the archives with in a similar fashion, using the file `PGPKEYS`

In each case an attacker can trivially circumvent the protection offered by using public key signatures by inserting their own key into this location. Due to its location, users will download this key and trust it and verify the signature using this key.

4.2 Key Compromise

One additional consideration in the evaluation of the trust of the signatures on software archives is the age of the keys. Older keys are typically of a smaller bit size and thus weaker than their newer counterparts. This is due mainly to the limitations of the software at the time of the key's creation. As shown in Figure 1, most keys are no more than 3 years old, but a significant number of the keys used to sign the archives examined in this study are 5 or more years old. Key age and size are related to their likelihood of their compromise by factorization by a determined adversary.

Similarily, when the sizes of the keys are examined, a strong trend towards the default settings is immediately appearant. Later versions of PGP and the gpg tool use 1024 bits as the default key size. The observed key lengths in bits are plotted in figure 2. When looking at recommendations by both schneier and rivest, 1024 is on the short end at the current time (2002) for security against even a modestly funded foe [6].

Because earlier versions of PGP had limitations on the sizes of keys they could generate, the year of key creation was examined in relation to the size of the key. As shown in figure 3, no strong correlation between the date of key generation and size in bits exists. In general, larger keys are found in the more recent years. Note that in every year found, 1024 bit keys are also in use.

The size of keys is a factor in the security of the key, and thus the signed archive. Recent advances in factoring of public keys [7] have caused some to abandon the reccomendations made by Schneier, Rivest, and others [6] in favor of larger keys [8].

Obviously the nature of the adversary must be taken into account. Software which is of interest to governments or large corporations to alter are under more serious threats than software which only interests individuals. Software of the former include cryptography products like IPsec and SSH implementations; the latter includes tools like Gnuplot and SMNP. It is safe to assume that factoring keys above 512 bits is still of interest only to dedicated researchers, large corporations, and governments.

4.3 Few Signatures in Use

By far the biggest issue uncovered was a large number of keys used to sign archives which were self signed or contain a low number of signatures by other parties. From the analyzed archives a total number of 93 unique keys were retrieved which contained 1971 signatures. This yields an average of 21 signatures per key. The number of signatures per key is plotted in figure

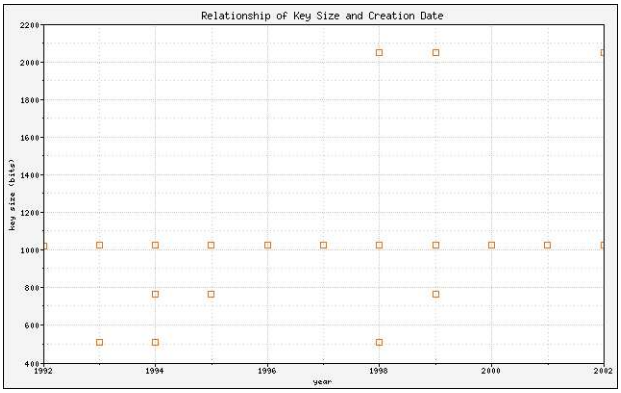

Figure 3: Correlating key age and size in bits.
The ages of the keys studied here were plotted against their sizes in bits. While a general trend of larger keys as the age decreases is observed, it is only a general trend. It is also interesting to observe that 1024 bit keys are present at every sample period. It is important to note that these keys are all in active use at this time.

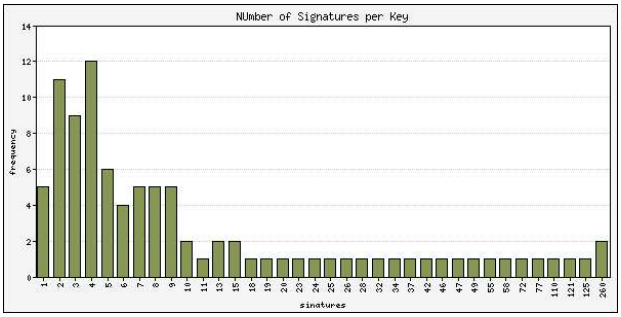

Figure 4: Signatures per key.
Using the keys downloaded to verify the software archives studied in this research, the number of signatures per key was measured and the resulting frequency plotted. Each key has at least one signature (from itself), with most keys having 4 signatures. Only a handful of keys have more than 12 signatures. note that this does not indicate the strength of the signatures, measured by their connectedness.

4.

At the maximum a number of 260 signatures were found on two keys used in this study, from Joost van Baal and Eduard Bloch, each involved in the Debian Linux project and avid initiators of key signing events. At the minimum, five keys were found to have only one signature. This signature was from the self signing of the key, which adds no security to the key. A total of 25 keys were found to be in use with 3 or fewer signatures on each key.

Notable keys which are in active use with 3 or fewer signatures include the official Scyld computing key (scyld.com), the Procmail distribution key, each with 3 signatures on each key (2 when the self signature is discounted), the FreeS/WAN 1.97 snapshot key, which has 2 signatures, and the XEmacs distribution key, which has only one signature from itself. In each case this limited number of signatures reduces the amount of trust in the archive due to the possibility of an introduction of a forged key. In the absence of trustworthy signatures, the veracity of the key cannot be understood.

4.4 Trust of the Signatures

It is important to note that while it is easy to obtain a large number of signatures for a public key, it is important that these keys be trustworthy and verifiable, as well. Of the 1971 signatures found in the keys used in this study, 1662 had unknown user ids. Subtracting the self signatures (93, one for each public key used in this research), this leaves 216 keys which had established user ids based soley on other archive signatures. While a key may have many signatures, they only add value when they are from known and trusted parties.

To examin the weight these signatures have, each of the keys used to sign the archives used in this study were mapped back to the largest "strong set" [9]. The strong set is a large set of approximately 53,000 keys which are referenced to eachother via signatures. The center of the set is considered to be the key with the minimum average distance between itself and any other key.

The 93 keys used in this study were analyzed using the strong set key analysis performed by Jason Harris on December 1, 2002 [10]. Of the 93 keys measured, 36 were unable to be mapped back into the strong set. Of the remaining 57 keys measured, an average distance between that key and any other key is slightly over 6 hops, with an average minimum of 4.2 hops and a maximum of 10.7 hops.

From this analysis we can deduce a metric of confidence in the key's validity. Keys which can be tied into the large strong set are connected to other keys in that set and are more likely to be valid keys. Isolated keys and sets occur when a narrow bridge between any cluster and the larger set is broken, such as through a revocation. The linkage to the large strong set, however, gives confidence that the key is indeed a correct key.

4.5 Expired Keys

An additional question which can be investigated from this sampling of software distributions is the prevalence of archives signed with expired keys. PGP has a method of granting a key a fixed lifetime, after which the key is invalid. This is typically used to prevent attacks from keys which have been broken during their lifetime.

The summary of statistics for this data set with regards to key expirations are as follows. Of the 93 keys obtained for this study, 16 had expiration date. Of those 16, 3 had expired by the time of their use in this study in August, 2002. When examined in light of the key ages, the difference between 16 keys and 93 keys is understandable, given that PGP 2.6.x users were unable to set key expiration dates. However, newer keys used for signing made with newer software frequently lack expieration dates.

The concept of an expiration date is a double edged sword. It helps in defeating attacks which break the key's encryption provided the expiration time is before the anticipated compromise could occur. On the other hand, it forces the generation and distribution, as well as an updating, of the signature material.

Expired keys are difficult to detect without examining for them explicitely. Early versions of the GnuPG tool often confused users with the results of expired keys, including attempts to use them and the nature of the error message generated. Furthermore, many versions of PGP implementations, including GnuPG and commercial PGP, differ in how they handle expired or altered keys. These interopability problems only worsen the dillema of properly handling keys.

An issue related to expired keys is revoked keys. Key revocations can be generated and uploaded to key servers. In the absence of frequent verification of keys against authoritative key servers, revoked keys may still be in widespread use. This is very similar to the problems faced by X.509 hierarchies with CRL (certificate revokation list) distribution. An update of the keyring in January, 2003, using the recently added GnuPG feature to automatically refresh the keyring showed only one key listed had been revoked.

5 Related Work

This research is not the first foray into studying the application of public key cryptography in the real world. A 1996 study into the trust model as it is implemented in PGP reveals that the weakest component of the PGP trust model is the trust assigned by a person to any key [11]. An additional study examined the security of the private key from the strength of the passphrase in use. While the study did not accumulate a large pool of passphases, the data it did accumulate enough data to note that passphrase security is a threat to the security of the system [12].

Ongoing research into the strength of the web of trust model used by the PGP protocol has been instrumental in illustrating the integrity of the system. This work, developed by Drew Streib and continued by Jason Harris, is an integral part of verifying the veracity of a key used by an archive [9, 10]. A related effort was undertaken with the Global Internet Trust Register [13]. Published in 1999, the Register is a printed list of PGP keys, allowing for an out-of-band review of key material. This is crucial in establishing the authenticity of a key.

Lastly, an examination of an attack tree against the PGP model shows that theft of the private key by other means (such as compromise of a workstation) is a more substantial threat than factoring the public keys [14].

6 Conclusions

This paper has examined the verification of over 2800 signed archives downloaded from various worldwide sites on the internet. While only 2 failures to properly verify were found, several other threats to the security of the signed distribution model were found and discussed. These weaknesses include a lack of trustworthy signatures on the keys, old or low strength keys, and poor distribution methods of the public key. While the system as a whole is not failing, key points where improvements can be made have been identified. Possible additional work includes an ongoing project to verify archives found on the internet using similar methods employed here, and key signing facilitation, especially to trusted PGP users.

7 Methods

To find signed archives to download and evaluate, the google search engine was used with the search terms 'tar.gz.sig' and 'tgz.sig'. These search terms were chosen as they are popular signature file extensions. This search yielded 261 sites and 400 unique sites and subdirectories to check, representing 2804 archives to download along with 2804 signatures. Due to duplicates and mirrors, this list represented 1426 unique archives to download and evaluate. The 'wget' utility was used to download the directories, which included the archive

and the signature file, to a local machine for evaluation.

A small tool was written which used the GnuPG tool, version 1.0.7, running on OpenBSD to verify the signatures. The directory tree was walked and the archives and signatures were compared, with the actions logged. If the key did not exist in the local public keyring, it was fetched from PGP.mit.edu and the verification process was repeated. This key server was chosen as it is a well established key server which contains most of the keys used in this study. Failures were examined manually.

Keys were examined after the verification process to gather statistics on the stated key size, key creation date, and number of signatures on each key. No attempt was made to verify the veracity of the stated key signatures on any key, and no trust web analysis was performed.

Graphs were generated using Gnuplot.

The complete list of sites and archives analyzed in this study, as well as the tools used to process the data, are all available in a separate document at `http://monkey.org/ jose/signed-archives/data/`.

8 Addendum

During the preparation of this paper, it was found that the distribution files for Sendmail were modified in a fashion similar to the files of openssh. More information is available from the CERT advisory on the subject [15]. Additionally, a modified distribution of the tools `tcpdump` and the `pcap` library were also found in November, 2002 [16].

9 Acknowledgements

The author would like to extend his gracious thanks to Rick Wash, Marius Eriksen, and Niels Provos, all of the University of Michigan, along with Jason Peel, Florian Kohl, Jeff Godin, and V. Alex Brennen for their insightful comments and advice during this study and the preparation of this report. The PGP key databases maintained by Jonathan McDowell, Drew Streib, and Jason Harris were useful in the analysis of the retrieved keys.

References

[1] irssi.org cracked -¿ irssi's configure backdoored for past two month. 2002. Available online at http://www.irssi.org/?page=news.

[2] Dug Song. Re: Trojan/backdoor in fragroute 1.2 source distribution. 2002. Available on the Internet at http://lwn.net/Articles/1479/?format=printable.

[3] Niels Provos. OpenSSH Security Advisory (adv.trojan). 2002. Available online at http://www.openssh.com/txt/trojan.adv.

[4] Patrick Feisthammel. Explanation of the web of trust of PGP. 2002. Available from http://www.rubin.ch/pgp/weboftrust.en.html.

[5] V. Alex Brennen. Strong Distribution HOWTO. 2002. Maintained at http://www.cryptnet.net/fdp/crypto/strong_distro.html.

[6] Bruce Schneier. *Applied Cryptography, Second Edition*. John Wiley and Sons, New York, USA, 1995.

[7] D. J. Bernstein. Circuits for integer factorization: a proposal. 2002. This proposal is archived on http://cr.yp.to/papers/nfscircuit.ps.

[8] Lucky Green. 1024-bit RSA keys in danger of compromise. 2002. This mailing list message is archived at http://tin.le.org/vault/security/encryption/rsa1024.html.

[9] Drew Streib. keyanalyze - Analysis of a large OpenPGP ring. 2002. The keyanalyze data and tools are available at the website http://dtype.org/keyanalyze/.

[10] Jason Harris. keyanalyze - Analysis of a large OpenPGP ring. 2002. The continuation of the keyanalyze data is available at http://keyserver.kjsl.com/ jharris/ka/ on a biweekly basis.

[11] Alfarez Abdul-Rahman. An Analysis of PGP's Trust Model. 1996. Available online at http://www.cs.ucl.ac.uk/staff/F.AbdulRahman/docs/pgptr

[12] Arnold G. Reinhold. Results of a Survey on PGP Pass Phrase Usage. 1995. This classic paper is archived online at http://world.std.com/ reinhold/passphrase.survey.asc.

[13] Ross J. Anderson (Editor), Bruno Crispo, Jong-Hyeon Lee, vacla Matyas, and Fabien A. P. Petitcolas. *The Global Internet Trust Register*. MIT Press, Boston, MA, 1999.

[14] Bruce Schneier. Attack Trees. *Doctor Dobb's Journal*, 1999. Archived at http://www.counterpane.com/attacktrees-ddj-ft.html.

[15] Chad Dougherty and Marty Lindner. CERT Advisory CA-2002-28 Trojan Horse Sendmail Distribution. 2002. This advisory is available online at http://www.cert.org/advisories/CA-2002-28.html.

[16] Roman Danyliw and Chad Dougherty. CERT Advisory CA-2002-30 Trojan Horse tcpdump and libpcap Distributions. 2002. The advisory and incident note is available at http://www.cert.org/advisories/CA-2002-30.html.