Security holes... Who cares?

Eric Rescorla
RTFM, Inc.
ekr@rtfm.com

Abstract

We report on an observational study of user response following the OpenSSL remote buffer overflows of July 2002 and the worm that exploited it in September 2002. Immediately after the publication of the bug and its subsequent fix we identified a set of vulnerable servers. In the weeks that followed we regularly probed each server to determine whether it had applied one of the relevant fixes. We report two primary results. First, we find that administrators are generally very slow to apply the fixes. Two weeks after the bug announcement, more than two thirds of servers were still vulnerable. Second, we identify several weak predictors of user response and find that the pattern differs in the period following the release of the bug and that following the release of the worm.

1 Introduction

It's widely believed that users are not particularly diligent about installing security fixes to their software. For instance, Netcraft [1] reports that 75% of users of SSL-ized versions of Apache had failed to upgrade their servers months after the Apache chunked encoding vulnerability [2] was announced. [1] Despite this belief, an enormous amount of effort has been put into the discovery, repair, and disclosure of security flaws in software. The implied rationale for all this work is that users who are informed about the security flaws of their software can upgrade as appropriate.

In this paper, we describe the results of direct measurement of deployment of fixes for the popular Open Source SSL/TLS toolkit OpenSSL, as installed in the mod_SSL package for the Apache Web server. For a number of reasons we would expect mod_SSL users to be better than average about installing security fixes:

- OpenSSL is security software and therefore its users clearly desire security.
- OpenSSL users are overwhelming UNIX users and UNIX users are widely believed to be more experienced in server administration than Windows users.
- Many popular operating systems (Linux, *BSD) have packages to make installing OpenSSL easier.
- We are studying the deployment of OpenSSL in servers which areparticularly vulnerable because they must be open to the Internet at all times.
- The flaw allowed an attacker to take over the entire Web server.

In spite of all these factors, our measurements show remarkably slow deployment. One week after the flaw was announced, only 23% of the servers under study had been fixed. Two weeks after, less than 1/3 had been fixed. At the time of release of the Slapper worm that exploited this vulnerability [3], almost 60% of servers were still vulnerable. Figure 1 shows the percentage of vulnerable servers for the period under study.

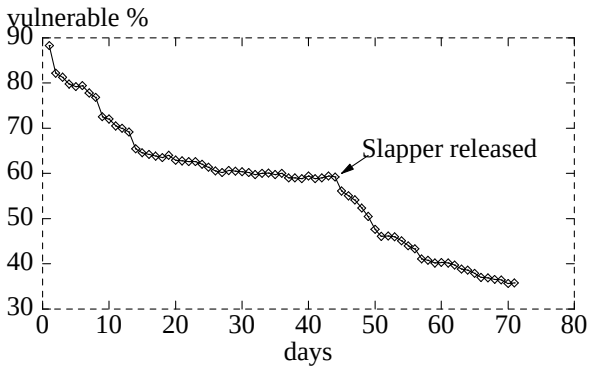

Figure 1 Vulnerable servers over time

2 Disclosure of the Vulnerabilities

On the morning of July 30, 2002, Ben Laurie, a member of the OpenSSL core team, sent an advisory entitled "OpenSSL Security Alert - Remote Buffer Overflows"[4] to a number of Internet mailing lists, including openssl-users and bugtraq. This announcement described the following flaws in OpenSSL:

1. The SSLv2 CLIENT-MASTER-KEY message was being improperly processed by servers. An overly-long message could be used to overrun a buffer on the heap. This bug was known to be exploitable.
2. An overlong SSLv3 SessionID value supplied by the server could be used to overrun a buffer on the client.
3. An overlong SSLv3 master key supplied to a server could cause overflow. This bug applied only to beta versions of OpenSSL 0.9.7.

1. Note that 25% of servers upgraded is a lower bound on fix deployment since a variety of countermeasures and patches for the Apache flaws were available.

4. Various buffers for ASCII representations of integers were too small on 64 bit platforms.
5. The ASN.1 parser could be confused by supplying it with certain invalid encodings.

The most important of these bugs were (1) and (2). Bug (1) would allow compromise of any OpenSSL server running SSLv2 (nearly all do) and bug (2) would allow compromise of any OpenSSL client running SSLv3 (nearly all do). The server bug was particularly serious because any attacker could connect directly to a vulnerable server and compromise it, whereas the client bug could only be exploited if the client would be induced to connect to the attacker's server. This is more difficult but by no means impossible. Since the server vulnerabilities were more serious and easier to measure we will be discussing them for the rest of the paper.

2.1 Dissemination

The wide popularity of OpenSSL led to extensive publicity almost immediately. The bug was widely reported in the press and within a day or two, operating system and application vendors began announcing versions of the advisory customized for their platform, along with instructions for upgrading. Figure 2 shows a timeline of major announcements.

July 30, 2002 10:08 GMT	Initial announcement
July 30, 2002 10:15 GMT	Patches for other versions released
July 30, 2002 11:08 GMT	OpenSSL 0.9.6e available for FTP
July 30, 2002 12:47 GMT	Debian announces[5]
July 30, 2002 13:00-12:00 GMT	Trustix, Engarde, Gentoo[6, 7, 8]
July 30, 2002 13:59 GMT	Vulnerability posted to Slashdot [9]
July 31, 2002 17:43 GMT	FreeBSD announces [10]
August 2, 2002 15:17 GMT	Apple announces[11]
August 2, 2002 17:33 GMT	NetBSD announces[12]
August 9, 2002 11:54 GMT	OpenSSL 0.9.6g released

Figure 2 Timeline of disclosure events

2.2 Scope of the Bug

At the time of the advisory there was a lot of confusion about which software was affected. Essentially, this confusion came in two forms:

1. Confusion about which SSL-using software was affected.
2. Confusion about which OpenSSL-using software was affected.

The first form of confusion stemmed from the fact that the flaws could be exploited by sending bogus SSL protocol messages and therefore the flaws were associated with SSL. No doubt the confusion was amplified by the discovery of problems with Internet Explorer's Basic Constraints processing [13] which was also associated in people's minds with SSL and all implementations thereof.

Although OpenSSL is the dominant free SSL implementation, many widely used SSL-using packages do not use OpenSSL. In particular, Internet Explorer, Internet Information Server, and Mozilla do not use OpenSSL and are therefore not vulnerable to this attack. (Despite being Open Source, Mozilla uses Netscape's home grown Netscape Security Services library rather than OpenSSL)

The second form of confusion resulted from the widespread use of OpenSSL as the cryptographic library for a wide variety of other applications. Since the flaws were primarily in the SSL protocol implementation, these applications were not affected. However, this fact was not widely understood and many vendors recommended upgrading not only OpenSSL but any application which depended on it. For instance, the OpenPKG advisory [14] recommended reinstalling OpenSSH, scanssh, and tcpdump, which used OpenSSL's cryptographic functions but not its SSL implementation or ASN.1 routines. It seems likely that some users who would otherwise have upgrade did not because they believed it to be a much larger job than it in fact was.

2.3 Round Two

The security changes made to OpenSSL included a number of assertions designed to check for various illegal protocol conditions. These conditions were not known to be exploitable but checking for them was clearly an improvement. Unfortunately, the assertions were structured explicitly as assertions and failure caused the program to exit. This made it rather easy to mount a DoS attack on an SSL server by feeding it invalid data and thus causing a crash.

In order to fix this bug, the OpenSSL team released OpenSSL 0.9.6f on August 8, which replaced the assertions with errors. Due to inadequate preparation time, there were a number of build problems with 0.9.6f and the team was forced to issue 0.9.6g to solve those problems. As of this writing, 0.9.6g was the latest stable version. The OpenSSL team did not issue patches for these problems but a number of operating system vendors did.

2.4 The Slapper Worm

On September 13th at 13:55 GMT, Fernando Nunes announced [3] that a worm had compromised his machine via the SSLv2 hole described here. The existence of this worm was independently verified and it was soon dubbed Slapper. The details [15] of the Slapper worm aren't important to understand, so we will simply summarize the important points.

Vulnerable Systems

Slapper randomly chooses IP addresses and then probes them in an attempt to determine the server version. When it finds a vulnerable server, it uses the SSLv2 buffer overflow to install itself on the target. Since each version of Apache requires a slightly different exploit, Slapper contains a table of version/memory offset pairs. If the version is not in the table, Slapper attempts to attack it anyway, using a default version guess, but this likely just creates a crash rather than a compromise, unless the guess happens to be correct. As of this writing, Slapper only attacks Linux machines, but it could be easily adapted for other platforms.

Worm Behavior

Once Slapper has compromised a target machine, it then does two things. First, it joins a peer-to-peer network of other worm instances. This allows the worm to be remotely controlled. The worm can then be used to DoS attack other systems or damage the compromised machine. In addition, the worm then looks for other machines to infect.

3 Countermeasures

As usual, the advisories admonished users to apply fixes as soon as possible. Users had four choices:

1. Turn off OpenSSL entirely.
2. Disable SSLv2 (server-only).
3. Upgrade to a newer fixed version.
4. Install one of the patches (supplied with the advisory).

Turning off SSL support isn't a realistic option for most sites but it's worth considering the other three options.

3.1 Disabling SSLv2

Turning off SSLv2 is by far the simplest possible countermeasure, which is effective on all 32-bit platforms. In both mod_SSL and ApacheSSL, a simple configuration directive will turn off all support for SSLv2. All the administrator needs to do is edit the configuration file and restart the server. Although administrators are sometimes concerned about being unable to serve SSLv2-only clients, SSLv3-capable clients have been available since 1996 and SSLv2-only clients are extremely rare. Therefore, the negative side effects of this countermeasure are quite minimal. [2]

3.2 Upgrading

The OpenSSL project currently maintains two source branches, the stable 0.9.6 branch and the beta-level 0.9.7 branch. Concurrent with the security alert, the OpenSSL team released versions 0.9.6e and 0.9.7-beta3. In both cases the releases were both source and binary-compatible with the previous releases in that branch, though not necessarily with other branches.

OpenSSL is extremely portable and so a source-level upgrade is quite easy. Assuming that the new version is compatible with the current version, the user merely needs to do:

```
./config
make
make install
```

In most cases, applications are dynamically linked against OpenSSL and so simply reinstalling OpenSSL is sufficient. Obviously, in the few cases where applications are statically linked they must be recompiled.

3.3 Patches

The OpenSSL project provided patches for all major versions of OpenSSL. We have test-applied them and they seem to work fine. In many cases, Operating System vendors also supplied patches for the versions they had shipped with their system.

3.4 Operating System Vendor Behavior

One of the primary channels for deployment of OpenSSL is via Operating System vendors. NetBSD, FreeBSD, and OpenBSD all provide OpenSSL as part

2. SSLv2-only servers have taken quite a long time to disappear, since a number of Netscape Enterprise server versions were SSLv2-only. This fact may explain the widespread concern over SSLV2-only clients.

of their base system. All the major Linux vendors provide packages (RPMs or .debs) for some version of OpenSSL. As a consequence, many users have never had the experience of building OpenSSL from source and expect to get their updates from their Operating System vendor. Figure 3 shows the updates provided by the major Open Source vendors.

Vendor	Update Type	Binary
Debian	Patch (deb)	Yes
Engarde	Patch (RPM)	Yes
OpenPKG	Patch (RPM)	No
SuSE	Patch (RPM)	Yes
Red Hat	Patch (RPM)	Yes
Mandrake	Patch (RPM)	Yes
Connectiva	Patch (deb)	Yes
Caldera	Upgrade (RPM)	Yes
OpenBSD Patch	CVS	No
NetBSD	Both (CVS)	No
FreeBSD	Patch (CVS)	No

Figure 3 Updates provided by various vendors

As noted above, all the major Open Source vendors were extremely proactive about delivering updates. In every case, updates were available within 2 days of the initial advisory. As should be apparent from Figure 4, updating on Linux was rather easier than updating on *BSD, since all of the *BSD updates required compilation, either of the base system or from the ports/packages collection.

4 Methodology

Our method is conceptually simple. We first assembled a list of SSL-capable servers and then periodically probed each server to determine whether or not it had deployed the fix, as well as what kind of fix (disabling SSLv2, patch, or upgrade) had been deployed.

4.1 Assembling the Corpus

It was first necessary to obtain a large list of SSL-capable servers. Although a number of different types of servers run SSL, we restricted ourselves to HTTPS (Web) servers because they are by far the most common and the easiest to find In addition, using only HTTPS servers allows us to have a more homogeneous sample and thus to draw tighter conclusions, at least for this limited domain.

Following the method of Murray [16] we built our list using a search engine. We randomly selected a list of 10,000 words from FreeBSD's `/usr/share/dict/words` file and then queried Google for each word using a search for "https and <word>" with the limit of hits per page set to 100. We then processed each page to find all the URLs on the page beginning with "https://" and built a list of all hosts which were potential SSL servers. We then contacted each server on this list and selected those which responded to an SSL query (with a 5 minute timeout) and which advertised that they used OpenSSL (thus limiting the sample size to mod_SSL and its derivatives since ApacheSSL does not advertise OpenSSL version). After suppressing duplicate IPs and servers whose administrators complained about being probed [3] the final sample size was 890 hosts. To ensure that we use the same server every time we sample by IP address, not DNS name.

4.2 Probing

Our task is to determine which of actions any given administrator has taken. To do this, we need three pieces of information:

1. What version number does the server advertise?
2. Does the server support SSLv2
3. Is the server vulnerable?

Given these pieces of information the decision procedure, shown in Figure 4, is relatively straightforward.

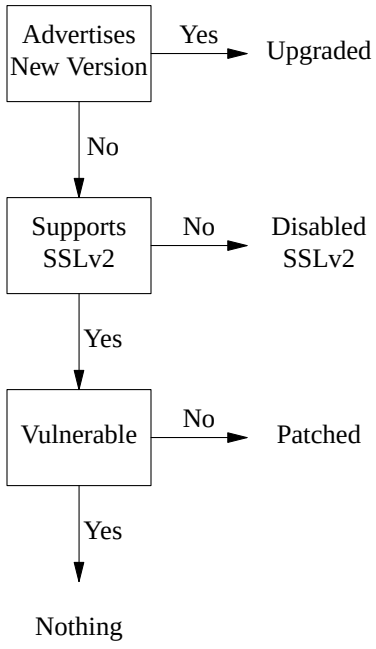

Figure 4 Determining what the administrator has done

3. n=2

Advertised Version

mod_SSL and its derivatives advertise the OpenSSL version in the HTTP `Server:` header line. It is therefore quite easy to determine what version of mod_SSL has been linked to the server. Thus, we can simply connect to the HTTPS server and issue a `HEAD` request. The server responds with an HTTP header containing the `Server:` field and hence the answer we desire:

```
Server: Apache/1.3.26 (Unix) mod_ssl/2.8.10 OpenSSL/0.9.6
```

SSLv2 Support

Determining whether the server supports SSLv2 is equally easy. We simply configure our client to offer only SSLv2 and then attempt to handshake with the server. A success means that the server supports SSLv2. A failure means it does not.

Detecting Vulnerable Servers

The patches provided by the OpenSSL project do not change the version number. As a consequence, we cannot simply examine at the advertised version number in order to determine whether or not they have been applied. However, due to the nature of the SSLv2 problem, it's still relatively straightforward to determine whether or not a server is vulnerable and therefore by excluding servers which advertise a newer version number identify whether or not a given server has been patched. [4]

The message used to exploit the OpenSSL SSLv2 problem is the CLIENT-MASTER-KEY message, which has the syntax:

```
char MSG-CLIENT-MASTER-KEY
char CIPHER-KIND[3]
char CLEAR-KEY-LENGTH-MSB
char CLEAR-KEY-LENGTH-LSB
char ENCRYPTED-KEY-LENGTH-MSB
char ENCRYPTED-KEY-LENGTH-LSB
char KEY-ARG-LENGTH-MSB
char KEY-ARG-LENGTH-LSB
char CLEAR-KEY-DATA[MSB<<8|LSB]
char ENCRYPTED-KEY-DATA[MSB<<8|LSB]
char KEY-ARG-DATA[MSB<<8|LSB]
```

4. A similar technique was described by Weimer [17] while this paper was in production. Weimer appears to have developed his technique based on analysis of the Slapper worm.

The first two fields are fixed length simply a message type followed by the cipher suite that the client has chosen. The rest of the message consists of the *clear key* (used in export mode), the *encrypted key*, which is simply a random value encrypted in the server's public key, and the *key arg*, which is used as the block cipher IV. In the CLIENT-MASTER-KEY message, the lengths of each field is specified first, followed by the concatenated values of all three fields. Thus, the last bytes of the message are the value of the key-arg field.

The source of the problem is that the key arg is a fixed-length buffer in the OpenSSL `session` structure. All of the block ciphers specified in SSLv2 have 64-bit block sizes and therefore the maximum size of the key arg is 8 bytes. When the CLIENT-MASTER-KEY message is read in by the function `get_client_master_key()`, this length is never checked and therefore an over-long key arg can overrun the buffer and cause a heap overflow. Since OpenSSL uses function pointers extensively, a heap overflow is easy to exploit. The fix for the problem is simply to check the length value of the key arg prior to copying it into the buffer, as shown in the code in Figure 5.

```
n2s(p,i); s->session->key_arg_length=i;
if(s->session->key_arg_length > SSL_MAX_KEY_ARG_LENGTH)
       {
       SSLerr(SSL_F_GET_CLIENT_MASTER_KEY,
              SSL_R_KEY_ARG_TOO_LONG);
       return -1;
       }
```

Figure 5 Length check in OpenSSL 0.9.6e

It's possible to determine whether OpenSSL has been patched by generating a CLIENT-MASTER-KEY which is one octet too long. It's easy to see that in a patched version, this falls afoul of the length check shown in Figure 5. The result is a handshake error.

In an unpatched version of OpenSSL, the server overruns the `key_arg` with whatever the extra byte is. However, as we shall now show, this overrun is harmless.

Once the overrun has occurred there are now two pieces of incorrect data:

1. The next field in the `session` structure is now damaged.
2. The `key_arg_length` is 9 instead of 8

Fortuitously, both of these changes are harmless. Assuming normal structure ordering the next field in the `session` structure is the `master_key_length` (with sparse field layout the overrun may damage empty data). This field is set at the end of

`get_client_master_key()` and is not used elsewhere in the function and so the damage goes unnoticed.

The `key_arg_length` field is never used elsewhere in OpenSSL. As each encryption algorithm knows its own block size, the `key_arg` buffer is passed directly into the encryption routines as an IV without a length attached. As a result, all the keying material remains the same and the handshake completes successfully. It's therefore easy to detect an unpatched OpenSSL server by simply using an overlong key arg and checking to see if the handshake completes.

4.3 Differentiating Fixed Versions

Although the OpenSSL team did not distribute patches for the DoS vulnerabilities, a number of vendors appear to have done so. If the fixes introduced with 0.9.6g had been restricted to fixing the DoS vulnerability, it would not have been possible to probe for these updates without damaging the servers in question. However, the updates also slightly modified the behavior of the server to the vulnerability probe described in Section 4.2. Figure 6 shows the overlong key-arg test found in 0.9.6g.

```
n2s(p,i); s->session->key_arg_length=i;
if(s->session->key_arg_length > SSL_MAX_KEY_ARG_LENGTH)
       {
       ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
       SSLerr(SSL_F_GET_CLIENT_MASTER_KEY, SSL_R_KEY_ARG_TOO_LONG);
       return -1;
       }
```

Figure 6 New key arg length check

Note the call to `ssl2_return_error()`. This generates an SSLv2 error message on the wire, whereas the previous check merely closed the connection. It is therefore possible to detect the newest round of fixes by the presence of this message.

4.4 Sampling Procedure

The primary work of sampling is done by two programs, `id-client` and `hstest`. `id-client` attempts the `HEAD` request described in Section 4.2 and `hstest` simply attempts a handshake and records whether it succeeds or fails. For historical reasons, both version detection and SSLv2 testing are done with `id-client` and patch detection is done with `hstest` . SSLv2 testing could be done just as well with `hstest`.

We sample four times daily at 6 hour intervals. We test version advertisement and SSLv2 support four times a day, at 6:00 AM, noon, 6:00 PM, and midnight Pacific. Because the patch detection creates a noticeable signature on the target server, which often annoys administrators, we test only twice a day, at 6:00 AM and 6:00 PM. In a number of cases we've encountered network or other problems during testing, in which case we've collected data as soon as possible thereafter. The hourly variance in performance is small enough that we believe this does not seriously affect the data.

5 Results

This section describes the data collected during the four weeks following the original announcement.

5.1 Baseline

In our first complete sample, taken at July 30, 16:43 GMT, our sample showed evidence of the deployment of 14 separate versions of OpenSSL. The vast majority of these versions were from the stable 0.9.6 branch. Figure 7 shows the distribution.

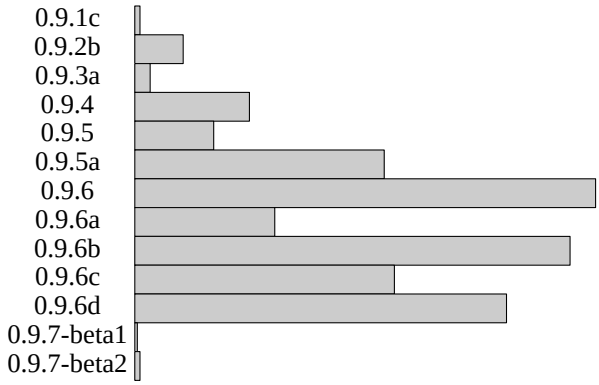

Figure 7 Version distribution at bug discovery

Note that version 0.9.6d was not the most popular version at bug discovery time.

5.2 Propagation of Fixes

The first question we want to ask is: how fast do administrators act to close security holes? Since we directly measure the vulnerability of a given server, this question is easy to answer, as shown in Figure 8. Figure 8 shows the percentage of servers in our sample each day that were vulnerable.

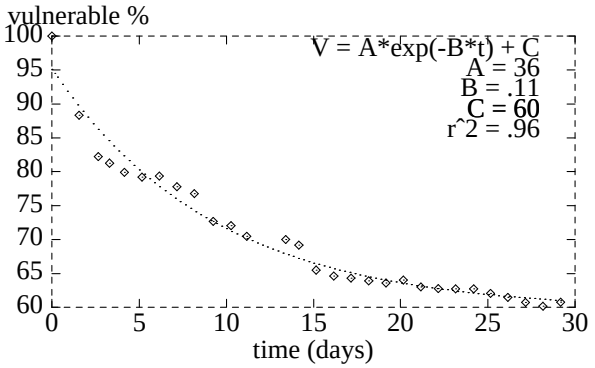

Figure 8 Vulnerable servers as a function of time

The obvious model for this data is an exponential decay curve. We have fitted this data to $vulnerable = Ae^{-Bt} + C$, producing the curve shown in Figure 8. Installation of security fixes is heavily front-loaded. A very large number of people upgraded immediately, with 16% of the sample population installed fixes within the first day and 20% within the first week. However, after that time, deployment rapidly falls off with the deployment curve going almost totally flat after two weeks.

As shown in Figure 8, roughly, 2/3 of the users who were going to install fixes had done so within two weeks. This suggests that the population is strongly bimodal, with roughly 35% of the population closely monitoring security updates and upgrading immediately and the remaining 60% taking no action of any kind.

Note that Figure 8 shows three big jumps, on days 0-1, days 6-7, and days 12-13. The first jump is a genuine mass upgrade event but the second two jumps are artifacts, reflecting deployment by two large hosting providers, SecureSites on August 7/8 and Verio on August 12/13.

Figure 8 slightly overestimates the number of patches that were applied, since only servers which answered and were vulnerable were counted. However, this effect is slight—during the measurement period no more than 35 servers were unavailable at any given time.

5.3 Day-by-day results

It's natural to wonder whether administrators are more likely to upgrade on one day than another. In order to explore this, we first need to control for the overall upgrading trendline, which is an exponential. In order to do so, we take the daily rate of change in vulnerable hosts as a percentage of the number of vulnerable hosts. So, for instance, if we have two samples ($time_0, hosts_0$) and ($time_1, hosts_1$), the normalized rate of change over that period is given by:

$$d\frac{hosts}{dt} = \frac{hosts_1 - hosts_0}{(time_1 - time_{1)} \cdot hosts_0}$$

Because our samples are over 24 periods, we now have the normalized daily rate of change. Samples are taken at 6:00 AM Pacific and so we arbitrarily assign the rate of change between day X and day $X + 1$ to day X. This produces some distortion because the assigned days do not exactly match days in North America, but of course this would be true of any dividing line we chose. Realistically, not much happens in North America before 9:00 AM Eastern so this dividing line produces a relatively accurate picture of what happens each day. Alternately, one can think of days as being measured in a time zone somewhere in the middle of the Pacific. The result is shown in Figure 9 as a scatter plot. Since the results span many weeks there are a large number of points corresponding to each day of the week. The horizontal line shows the average rate of change for each day.

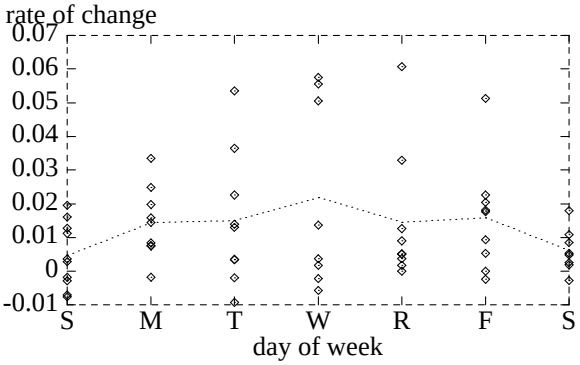

Figure 9 Daily rates of change

As Figure 9 shows, qualitatively more upgrading happens during the North American workweek than on the weekends. There is probably some effect here from the release time of the advisories (Tuesday morning and Friday morning), because we would expect the first day or two afterwards to be especially active. However, the general trend of all the data points is higher during the week than on the weekend. Note that this is a qualitative result only.

With the exception of the two mass upgrades mentioned in Section 5.2, there does not appear to be any significant difference in the amount of upgrading that happens during the North American daytime and the North American evening. However, it's possible that our measurements are not sensitive enough to distinguish evening from morning due to the sampling frequency and times.

5.4 Who upgrades?

Finally, let's consider the question of which class of users choose to deploy fixes. This question is not of completely academic interest. It would be convenient for vendors (if not altogether to the public good) if some readily identifiable group of users deployed and the rest did not, since they could then either ignore or cater to specific segments. (This information would be useful to attackers as well, one supposes.)

In this section, we consider a number of potential predictors of administrator response. In general, we might expect that a server which is actively maintained would be more likely to be updated in response to security problems. We therefore consider a number of measures of how actively maintained the server is, including whether software versions were up to date, whether it is hosted by a hosted service provider (HSP)—which presumably maintains it more actively—and a number of metrics of how "live" the web site is. We also wanted to assess the effect of TLD (reported to be relevant by [18].)

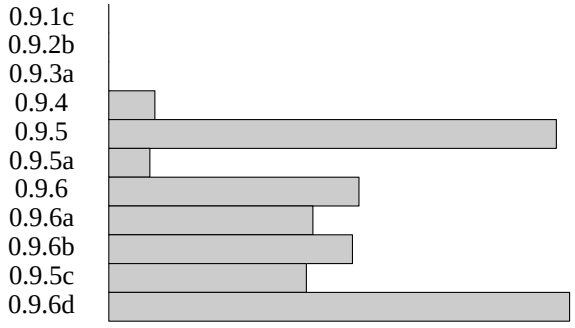

Figure 10 Fix deployment by original version

Version Effects

The intuitively obvious predictor of whether users will apply fixes is whether they already are up to date. Figure 10 shows the percentage of servers initially running each version who applied fixes to their servers by the end of the initial 28-day window. With the exception of servers initially running version 0.9.5, we see a relatively consistent pattern: servers which were initially are fixed at the highest rate (70%). Servers which are running other 0.9.6 versions are fixed at a rate of about 30-40%. Less than 10% of servers 0.9.5 and 0.9.4 variants were fixed, and no servers running versions < 0.9.4 were fixed. The overrepresentation of running 0.9.5 is a sampling artifact— WHOIS lookups on the relevant IPs show all the servers to be operated by Cybergate, so their results are not independent.

Analysis bears out this qualitative assessment. Figure 11 shows the results of a logistic regression [19, 20] predicting the occurrence of upgrading. The explanatory variables are the initial versions of OpenSSL and Apache, with the Apache version being a dichotomous variable of either up to date (post the chunked buffer overflow) or old. The regression was performed with R [21] using the Design [22] library.

Coefficient	Meaning
χ_1	OpenSSL 0.9.6x (x<d)
χ_2	OpenSSL 0.9.6d
χ_3	Apache Current
χ_4	OpenSSL 0.9.6x (x<d) * Apache Current
χ_5	OpenSSL 0.9.6d * Apache Current

Variable	Coefficient	Standard Error	Z	P value	Odds Ratio	95% CI
Intercept	-1.52	0.217	-7.03	-	-	-
χ_1	0.335	0.261	1.29	0.199	1.40	0.839-2.33
χ_2	1.52	0.740	2.06	0.040	4.58	1.07-19.5
χ_3	-0.782	0.567	-1.38	0.168	0.458	0.151-1.39
χ_4	1.82	0.602	3.02	0.003	6.14	1.89-20.0
χ_5	1.54	0.925	1.67	0.096	4.67	0.762-28.6

Figure 11 Logistic regression model for updating behavior

As expected, we see that initial version is a significant predictor of updating behavior. Servers which were running 0.9.6d when the bug was released were more likely (odds ratio=4.58; P=0.04; 95% CI= 1.07-19.5), regardless of what Apache version they were running. However, the Apache version is a significant effect modifier when the initial OpenSSL version is 0.9.6x (x<d), even though Apache version by itself is not significant. We interpret this result as follows: The latest Apache versions contain fixes for a remotely exploitable hole for which exploits [2] were circulating. Thus, Apache version is a discriminator between people who just happen to be up to date and people who are consciously remaining up to date on security fixes (in this case for Apache)

Hosting Services

Unfortunately, the results obtained in the previous section are not entirely adequate. The problem is that the samples are not truly independent, since a fair number of the sites in question are operated by large hosting service providers. In general, we would expect all the machines operated by a given provider or administrator to be upgraded simultaneously, regardless of what software versions they were running. Discovering the operator of any given server would require contacting each administrator and asking relatively detailed questions. We considered this prohibitive.

Instead, we used WHOIS [23] net blocks as a proxy. We first interrogated the ARIN database. In cases where ARIN reported that the net block was administered by a different registry such as APNIC or RIPE we then interrogated that registry's database. We considered that any servers which belonged to the same net block were operated by the same provider.

This assignment is necessarily somewhat rough. First, some providers do not report net blocks which are assigned to customers as delegated. The result will be that some hosts which are actually operated by different administrators will appear to be operated by the same provider. Second, some large ISPs (e.g. Verio) own multiple net blocks with different names, perhaps due to acquisitions. We made no attempt to determine whether the machines in these blocks were separately administered. Finally, some providers provide both managed and unmanaged service, in which case some machines in a managed facility may actually be operated by a customer.

Nevertheless, there is quite a noticeable effect of HSP size on updating behavior. Figure 12 shows the vulnerability percentage for some somewhat arbitrarily chosen intervals of HSP sizes.

HSP Size (Hosts)	% Vulnerable	# Hosts
1-4	0.71	673
5-15	0.50	46
15-30	0.33	69
30-100 (Verio)	0.15	69

Note: only hosts for which data was available at day 27 are listed here.

Figure 12 Upgrading by HSP size

We attempted to directly account for the effect of HSP size via logistic regression but were not able to achieve good fits. Reduced models did not have satisfactory fit statistics and the saturated model produces absurd coefficients due to a number of empty cells. These problems suggest that HSPs behave enough differently from ordinary administrators that it's worth treating them separately. Accordingly, we performed a new analysis with only the "independent" hosts—those with $HSPSIZE = 1$ (n=518). This yields an excellent fit, as shown in Figure 13

With the effect of HSPs removed, we now see that all three predictors are significant. In other words, having up to date or even modestly recent version of OpenSSL and Apache was a significant predictor that the server will be updated to fix this bug. Note that we do not show interaction terms between OpenSSL and Apache versions in this fit. Likelihood Ratio testing indicated that those terms did not significantly improve the quality of the fit. The results shown in Figure 13 are quite robust with regard to our classification of whether the server is operated by an HSP. We achieved good fits (P=.74) even when the breakpoint was set as high as 15 servers in a net block.

Variable	Coeffi-cient	Standard Error	Z	P value	Odds Ratio	95% CI
Intercept	-2.40	0.325	-7.38	-	-	-
χ_1	1.04	0.350	2.96	0.003	2.82	1.42-5.60
χ_2	1.49	0.445	3.35	0.001	4.44	1.86-10.6
χ_3	0.873	0.244	3.57	0.000	2.39	1.48-3.86

Hosmer-le Cessie Test: P=.56

Figure 13 Updating behavior of independent servers

By contrast, attempts to fit the behavior of HSP-hosted servers (with either breakpoint 1 or 15) were unsuccessful, for three reasons. First, since HSPs were comparatively responsive, there's less variation to account for and what was seen wasn't consistent across the factors under study, as shown in the contingency table in Figure 14. Second, the number of hosts in this stratum was somehat maller (n=373). As a consequence of these two factors, there were a number of empty cells in the contingency table, leading to misleadingly high coefficients. Finally, a single HSP (Alabanza) failed to respond at all by day 27. We conclude that the primary source of inter-HSP variation is some factor which is not tightly related to deployed software version.

	OpenSSL Status		
Apache Status	Downrev	0.9.6x (x<d)	0.9.6d
Downrev	.61	.71	.5
Up-to-date	1	.41	.18

Figure 14 HSP vulnerability by Apache and OpenSSL version

The Effect of TLD

Moore et. al. [18] report that TLD was a significant predictor of upgrading behavior in response to the Code Red worm. In order to test for this effect we took our day-27 results and filtered out all servers corresponding to TLDs with n<10. The remaining TLDs were: .au, .ca, .com, .de, .edu, .gov, .net, .org., and .uk) The chi-squared test provides no evidence of this effect ($\chi^2 = 2.665$; df=7; p=0.914). However, the sample size in that study was larger and therefore may have more resolving power.

Some Other Irrelevant Factors

In the search for other predictors, we also examined a number of factors. These included:

- Certificate status (valid, self-signed, etc.) — as a predictor of whether the site was "real" or not.
- Website "last-modified date" (only available for 418 sites) —as a predictor of whether the site was being actively maintained.

Neither factor showed any significant predictive value.

5.5 Upgrade or Patch?

Downrev versions are a perrenial source of difficulty for software vendors. Even for vendors, such, as OpenSSL, who do not make money off of upgrades, a substantial amount of time is spent providing support for older versions. Accordingly, vendors would very much like users to upgrade. Users, naturally, resist upgrading because it is disruptive. Upgrading to fix this kind of security problem, which is inherently a very small change to the code, is especially painful if it requires installing a completely new version with the attendant problems of interface instability and new bugs.

Since the OpenSSL team released both patches and a new version, an obvious question to ask is: did users primarily patch their existing code or did they primarily install a new version? Figure 15 shows the installation of both patches and upgrades during the period under study.

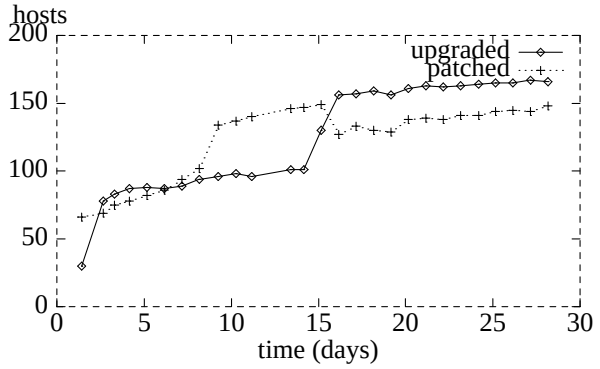

Figure 15 Propagation of upgrades and patches

As Figure 15 shows, patching and upgrading are about equally popular. This isn't surprising since, as described in Section 3.4, a large number of vendors shipped patches to the OpenSSL versions they already shipped rather than upgrades to 0.9.6e.

The data shown in Figure 16 shows three additional interesting features. First, in the day immediately after the announcement, vastly more users chose to patch than upgrade. We suspect two causes for this behavior. First, the patches were released an hour or two earlier than the 0.9.6e and so a really diligent administrator might have chosen to protect himself when only the patches were available. Our data is not fine-grained enough to test this hypothesis. The second possible cause is that administrators were in a hurry to deploy a stopgap fix and that patches were easier to deploy because they were less disruptive.

This hypothesis is somewhat borne out by the data between days 12 and 15 in which the number of fixed servers upgraded servers increased while the number of patched servers declined. What happened here is that a number of administrators first deployed patches and later upgraded to 0.9.6e.

In order to study this issue, we again built a logistic model for the patch/upgrade choice (using Upgrade as the event of interest). As before, we first stratified the data on *ISPSIZE* = 1. This allowed us to obtain an adequate fit for independent servers, which is shown in Figure 16. Note that the model here does not include an interaction between the OpenSSL version and Apache version since an adequate fit was obtained without that term.

Event: Upgrade

Variable	Coefficient	Standard Error	Z	P value	Odds Ratio	95% CI
Intercept	-0.250	0.650	-0.38	-	-	-
χ_1	-1.28	0.698	-1.84	0.066	0.278	0.071-1.09
χ_2	-0.166	0.818	-0.20	0.839	0.847	0.171-4.21
χ_3	1.26	0.484	2.61	0.009	3.53	1.37-9.13

Hosmer-le Cessie Test: P=0.92

Figure 16 Logistic regression model for upgrade/patch decision

The results here are relatively easy to interpret: users who are up to date on Apache are more likely to upgrade than apply patches. OpenSSL version is not a significant predictor. It's not clear why Apache version would be an important predictor when OpenSSL version is not. One possibility is that some subset of users respond to security problems by upgrading to the latest version. That subset would obviously have up to date versions of Apache and then would subsequently upgrade their OpenSSL versions in response to a vulnerability. One test of this hypothesis would be to measure the versions of other software deployed on the machines in our sample.

We can see a similar situation with HSPs. Of the 193 servers with *HSPSIZE* > 1 that responded to this vulnerability, 113 (58%) upgraded to newer versions of OpenSSL. There were too many empty cells to perform an adequate logistic fit but Figure 17 tells the story

5. In addition, we were unable to fit the saturated model because there were no servers with (OpenSSL 0.9.6d and Apache downrev).

clearly enough. Qualitatively, servers with downrev versions of Apache were more likely to be running downrev versions of OpenSSL and more likely to patch. Servers with uprev versions of Apache were more likely to be running OpenSSL 0.9.6 and more likely to upgrade. Servers running OpenSSL 0.9.6d were more likely to upgrade. There is no reason to doubt the obvious interpretation here—it's vastly easier to upgrade if you're already more or less up to date.

Apache	Downrev			Up-to-date		
OpenSSL	Downrev	0.9.6x (x<d)	0.9.6d	Downrev	0.9.6x (x<d)	0.9.6d
Patched	19	21	0	0	31	9
Upgraded	0	2	4	0	22	55

Figure 17 Upgrade/patch decision for HSPs

5.6 SSLv2 Not Disabled

Note that we have not discussed hosts which disabled SSLv2. At no time during the selected period did more than 10 servers which supported SSLv3 disable SSLv2 and only 5 consistently appeared to have done so. It's quite possible that the remainder of hosts which appeared to have disabled SSLv2 were merely attributable to measurement error.

This is a surprising result since this countermeasure is extremely easy to deploy and completely removed the vulnerability, intuitively one would expect administrators looking for a stopgap to employ it first and only later upgrade. However, essentially no administrators did so. This could result from a number of causes, but we believe that the most likely one is simply that administrators didn't understand that it was available. Although the original advisory mentioned that only SSLv2-capable systems were vulnerable, a large number (90+ %) of the vendor advisories did not. Moreover, as described in Section 2.2, there was widespread confusion about the scope of the problem, even among security experts who had read the original advisory. Therefore, it seems likely that administrators may simply have believed that fixing their implementation was their only option. We have no way of directly testing this hypothesis with our current data set. However, if a similar vulnerability arises with an immediate workaround but no fix we could examine how many users apply the workaround.

5.7 Deployment of Version 0.9.6g

As described in Section 2.3, the rush to deploy fixes for the initial round of bugs resulted in an incomplete fix with some DoS vulnerabilities. The fixes for these vulnerabilities were deployed in version 0.9.6g. However, deployment of version 0.9.6g was spotty. At day 27, the end of the survey period, less than 5% of the servers studied were version 0.9.6g. At the end of the study period, 173 (19%) of the servers were 0.9.6g and 121 (14%) were 0.9.6e. It is not completely clear why this is the case, but there were likely several contributing factors:

- The announcement was not as widely disseminated as the original announcement.
- The problems were not as serious (DoS only).
- Not all vendors released updates for the problems discovered in the first round of patches.

In general, it appears that users who upgraded to 0.9.6e did not upgrade again to 0.9.6g.

5.8 Response to Worm Announcement

The announcement of the Slapper worm initiated a new round of upgrades and patches. Figure 18 shows the data for the first two weeks after the announcement of the worm. As before, we have fitted the data to an exponential ($r^2 = .99$).

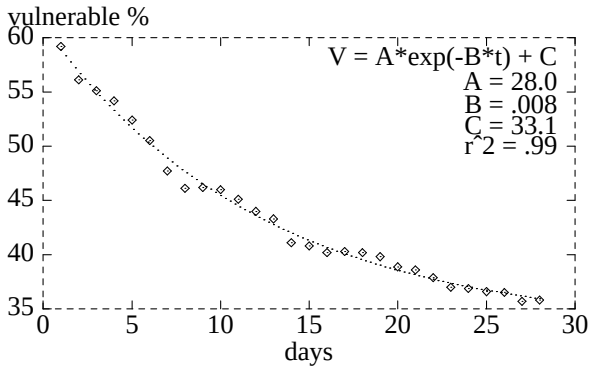

Figure 18 Upgrading after Slapper announcement

This second round of updates actually requires some explanation. Why do people who didn't upgrade the first time upgrade when the worm was announced? One possibility is simply that some administrators had not heard of the original bug. However, since the worm was announced through the same channels, this doesn't seem likely to account for the entire difference. A more likely explanation is that administrators have explicitly or implicitly adopted a strategy of only upgrading when an exploit is available, rather than merely when a bug is announced. As Beattie et al. [24] point out, it's often undesirable to update immediately, since patches are

flawed. Waiting until an exploit has been released therefore seems like a reasonable[6] wait-and-see attitude.

In order to study this, we built a logistic regression model. As before, we focussed on independent sites by removing all sites from net blocks containing more than one host. The results are shown in Figure 19. Note that the fit here is not particularly good, as shown by the low P value (P = .13). However, the saturated model does not have a significantly better log likelihood (P > .10) and none of the interaction terms are significant. However, as before, we find that the fit parameters are stable when we adjust the stratification point. In fact, if we use 15 as our cutoff, the parameters are qualitatively similar but the fit is far better (P = .94).

Variable	Coefficient	Standard Error	Z	P value	Odds Ratio	95% CI
Intercept	-1.358	0.256	-5.31	-	-	-
χ_1	0.647	0.296	2.18	0.029	1.93	1.07-3.45
χ_2	0.261	0.484	0.539	0.590	1.30	0.502-3.36
χ_3	0.897	0.270	3.32	0.001	2.45	1.44-4.16

Hosmer-le Cessie Test: P=.13

Figure 19 Logistic regression model for post-worm updating

As Figure 19 shows, OpenSSL 0.9.6x (x<d) is a significant but small predictor for post-worm response (odds ratio=2.00; P = 0.029; 95% CI=1.07-3.41). Apache up-to-date status is a somewhat better predictor (odds ratio=2.45; P=0.000; 95% CI=1.44-4.16). Note, however, that OpenSSL 0.9.6d is not a significant predictor. We interpret this result as follows: any administrator who was staying up to date on OpenSSL updated when the bug was announced. The remaining administrators respond to exploitable bugs. Since Apache status is a signal for whether or not administrators respond to public exploits, it therefore is a factor in predicting whether or not administrators will upgrade after release of the worm.

The implication of this analysis is that there are three roughly equals-sized classes of users.

- Users who respond immediately upon the release of security holes.
- Users who respond when exploits for holes start to circulate.
- Users who do not respond at all.

In order to confirm this hypothesis, one would need to do a horizontal study of updating behavior for a wide number of independent packages with variation in the known types of security flaw. A (non-random) sample could potentially be found by examining other software present on the systems in this sample.

Finally, we turn to the behavior of HSPs subsequent to the release of the worm. As mentioned previously, HSPs were in general very responsive to the initial disclosure of this bug. The single HSP that did not respond at all to the initial disclosure (Alabanza) responded completely upon release of the worm. This single HSP has a large effect on our estimates of which variables are significant. Without Alabanza included in the fit, the results are qualitatively similar to those shown in Figure 20. With Alabanza included in the fit, OpenSSL version is no longer significant independently but acts as a negative effect modifier for Apache version. Since Apache version remains a strong predictor, we have good reason to conclude that it is relevant, but we are unable to draw firm conclusions about the effect of OpenSSL version.

6 Policy Implications

Since this paper reports on a single disclosure event, we should be wary of drawing firm conclusions. Nevertheless, if we take this event as representative, administrator behavior in this case suggests some appropriate methods for managing future vulnerabilities.

6.1 Timing of Disclosure

As we saw in Section 5.2, the deployment of fixes is very strongly frontloaded. In both rounds of upgrading, nearly all of the administrators who eventually fixed their servers had done so within two weeks of the announcement of the vulnerability. However, a substantial fraction of those administrators did not upgrade within the first week. These two results have implications for the timing of bug disclosure.

Delayed Full Disclosure

Until recently it used to be quite common for vendors and coordination centers to provide a limited advisory that described the nature of the vulnerability in general terms and notified users of the availability of a fix. In some cases the vulnerability was never fully disclosed. In others, the vendors would wait a "decent interval" to allow users to upgrade. We are now in a position to assess this practice.

Since many users never upgrade we can easily see that it is never "safe" to release all the details of a vulnerability. There will always be a large number of users who remain vulnerable at any given time. However, it is

6. Note that Beattie et al. suggest that applying patches either 10 or 30 days after release produces the optimal shakeout/vulnerability tradeoff. Our data doesn't show any sign that users behave this way, however.

similarly clear that a long delay between limited disclosure and full disclosure serves no useful purpose. Since essentially all users who are going to upgrade do so quickly, full disclosure should take place within a month or not at all.

We take no position on the relative merits of full disclosure versus limited disclosure in general, except to note the obvious fact that full disclosure necessarily leaves a large number of users vulnerable. On the other hand, it is often argued that full disclosure encourages users to upgrade, which is no doubt a good thing. Whether full disclosure has benefits that outweight these costs is not a matter for discussion here. Obviously, this tension is far weaker in the case of Open Source software, since the deployment of patches almost inevitably provides enough information for an attacker to discover the nature of the vulnerability.

Disclosure Before Fixes Are Available

As we saw in Section 5.2, the users who are going to upgrade are very quick to do so. As a consequence, disclosing vulnerabilities before fixes are available results in a very high marginal cost to those users, since those users are vulnerable for the period between announcement and release of the patch—whereas if disclosure were withheld they would not be. The negligible use of non-fix countermeasures seen in Section 5.6 suggests that the availability of workarounds does not reduce this marginal cost. If at all possible, vendors should be given time to develop and deploy fixes.

6.2 What Fixes to Develop

Obviously, vendors would prefer to minimize the number of versions they support. In the best case, all users would upgrade to the latest version. Even if this is not possible, it would be desirable to issue patches only for the most recent versions. Unfortunately, as we saw in Section 5.5, a substantial fraction of users across all versions preferred patches to upgrading. We don't have enough data to determine whether or not those users would have upgraded were patches not available, but it seems likely that many would not have. By contrast, the value of devising and deploying workarounds seems quite minimal, given that essentially no users chose to deploy even the most minimal form of workaround.

6.3 Get it Right the First Time

It is universally agreed that OpenSSL 0.9.6g is superior to 0.9.6e. Nevertheless, at day 27, deployment of 0.9.6e was vastly greater than 0.9.6e for the simple reason that 0.9.6e got there first and solved the most pressing security problem: the remote buffer overflow. However, had 0.9.6g been released instead of 0.9.6e the DoS problems would be fixed as well. The rush to release 0.9.6e is more than understandable in view of the seriousness of the bugs. Nevertheless, the result was not quite optimal. Note, however, that users who upgraded after the worm was announced naturally installed 0.9.6g. The important lesson here is that users have a limited appetite for security fixes and that vendors who wish to deploy all their fixes should ensure that their first bugfix release includes them all.

6.4 The Impact of Vulnerable Servers

The impact of Web server compromise is obviously quite substantial. Aside from the high probability that the attacker will leverage the compromised server into administrator privileges, mere control of the server itself is dangerous. The attacker can vandalize the site, or, worse yet, recover the server's private key. The lack of reasonable revocation mechanisms means that private key compromise is catastrophically bad for the site in question, since it generally allows the attacker to decrypt any connections which he can sniff using that key, using a tool such as ssldump [25] The private key is generally stored in the clear in the server's memory and so recovering the private key once the server is compprompised is trivial and does not require achieving root access.

One common question that people seem to have is whether the residually vulnerable servers may not simply be defunct servers. This does not seem to be the case. Although we have not done a thorough survey, the list of vulnerable servers as of this writing includes a number of "prestige" sites, including Universities, government agencies, and well known ISPs. One interesting avenue of research would be to correlate blind user ratings of site "liveness" against vulnerability.

Even if the vulnerable sites were defunct, this would still be a security problem for the rest of the network, because compromised machines can be used as an attack platform. Recall that the Slapper worm does exactly this, turning the victim machine into a zombie for mounting DDoS attacks. Thus, it's critical for everyone for vulnerable machines to be fixed. This risk is particularly great, since, as shown in 5.4, independently operated servers are less likely to upgrade, and such servers are the most difficult for victim sites to have shut down. Potential measures along these lines would be for ISPs to filter un-patched servers. Dick [26] suggests a cleverer though perhaps more difficult option, which is for sites likely to be vulnerable to DDoS to pay other sites to upgrade.

6.5 Understanding HSP Policy

As we have seen, servers operated by HSPs generally are more responsive than independent servers. However, we have also seen that all HSPs are equally responsive. Accordingly, users who care about security should investigate their HSP's upgrading policy and (if appropriate) insist that the HSP provide response guarantees.

7 Sources of Error

In any survey of this type, there are a number of potential source of error. This section describes the known sources of error and attempts to assess their severity.

7.1 Availability Bias

Our sampling procedure is not guaranteed to give a completely random sample. In particular, it favors sites which advertise in Google. This creates two known sorts of bias:

- Bias against private sites which do not advertise.
- Bias against large sites which restrict searching.

These forms of bias act in opposition. The first eliminates some small sites which are unreachable via links. One might hypothesize that such sites are less well maintained and therefore less likely to respond to security vulnerabilities. The second form of bias eliminates sites which one might hypothesize were well managed and therefore more likely to respond. The current data do not allow us to evaluate the extent of either form of bias.

The extent to which this sort of sampling bias influences our results depends on what sort of conclusions you wish to draw. If one wishes to draw conclusions about the probability that a web site one accesses will have been compromised, this form of sampling is good. If one wishes to estimate the percentage of potential "zombie" machines available for DDoS, then it is possible that our sample is misleading. One way to cross-check would be to sample IPs randomly as common worms do. We have not done so.

7.2 Network Problems

This sort of experiment is subject to two forms of network failure. In the first, the network to our sampling machine fails and we are therefore unable to take data during the outage. In such cases, we resample after the outage. No such outage lasted more than 4 hours and so we do not believe that this has an appreciable effect on our data.

The second sort of outage is more troubling, since there is no reason to believe that unavailability of hosts is independent of other properties. However, since no more than 6% of hosts became unavailable during the sample period, the bias is relatively small compared to the size of the effects we're measuring.

7.3 Measurement Error

Remote probing is subject to several forms of measurement error:

- False version numbers.
- Load balancing.
- Acceleration

False version numbers

It's possible, though inconvenient, to modify one's OpenSSL to advertise a different version number. Thus, one could (for instance) advertise OpenSSL 0.9.6e while actually running OpenSSL 0.9.6d. We do not believe that this is widely done. Anyone with the expertise to make such modifications could quite easily apply patches.

Load Balancing

Suppose an operator runs more than one server behind a load balancer. For some reason or other, one machine is fixed and the other is not. In this case, we will get inconsistent results depending on the load balancer. In our tests, about 10 machines had repeated flip-flops between fixed and unfixed.

Accelerating Proxies

It's quite common to use an SSL reverse proxy [27] to offload SSL processing from a server. However, since the proxy is otherwise transparent, this means that the advertised OpenSSL version (from the server) might not match the real SSL implementation (whatever is on the proxy. Since most such proxies are based on OpenSSL, this might mean the proxy was vulnerable even though the server advertised a non-vulnerable version. Our data shows a small number of hosts (<10) which advertise non-vulnerable versions of OpenSSL but appear under probing to be vulnerable. Careful evaluation of the OpenSSL code indicates that the probes must generate an error with any repaired SSL version.

Therefore, we believe that this effect is responsible for the anomalies in question.

Block Ciphers versus Stream Ciphers

Subsequent to the survey we discovered a possible source of false negatives—`hstest` works by using a key-arg one byte longer than it should be. This causes an overrun when a block cipher is used but not when a stream cipher is used. Thus, if we negotiate a stream cipher, then the handshake will succeed. However, this is not a significant source of false negatives, for two reasons:

1. In SSLv2 the client chooses the cipher suite from a list of those acceptable to the server. Our client prefers block ciphers.
2. Nearly all servers support at least one block cipher.

Double-checking with a client programmed to support only block ciphers yields essentially identical results (+/- 1 server). Therefore, we do not believe that this is a significant source of error.

7.4 Analytical Issues

Sample Size

The relatively small sample size (n=890) presented some analytical challenges. In some cases, were unable to fully fit parameters of interest because the corresponding cells in the contingency table were empty, leading to absurd results. A larger sample would presumably have fewer completely empty cells and be more amenable to analysis.

Server Independence

As noted in Section 5.4, not all of our samples are completely independent. We adjusted for this by stratifying the sample into "independent servers" and HSPs. However, this stratification is necessarily somewhat imprecise. However, the fact that which predictors are significant is relatively insensitive to where the exact boundary is placed suggests that the stratification is good enough to draw reasonable conclusions about relevant factors. Confirmation of these results could be achieved by directly contacting server administrators and determining exactly who is responsible for which hosts.

8 Related Work

The general shape of the upgrading curve has been observed by previous authors. The closest related work is by Provos and Honeyman [28] who measured the deployment of OpenSSH versions subsequent to the release of the SSH CRC vulnerability [29]. The authors found a somewhat lower asymptotic (20%) upgrade rate, but it's not clear how reliable it is since in some cases they notified the hosts under study. Cheswick et al. [30] report similar results for response to a vulnerability in BIND. This work differs from previous work in three important respects. Finally, Moore et al. [18] report on user upgrading behavior in response to the Code-Red worm.

This work differs from previous work in three important respects. First, we follow the trajectory of hosts from release of a bug through release of a worm that exploits the bug. Previous work only measured one phase or the other of upgrading. This allows us to accurately compare response to different security stimuli. Second, because we are directly probing for vulnerabilities rather than merely version numbers, we are able to determine when servers have been patched or countermeasures have been applied as well as whether they have been upgraded. As Section 5.5 shows, a significant number of administrators opt to patch and therefore it's important to measure patch rate when patches are available.

Finally, since this is a longitudinal study of a specific set of hosts, we are able to characterize the various factors that predict administrator responses. We are not aware of previous work that addresses this issue.

	Initial Release		Post Worm	Upgrade/Patch
Factor	Overall	Independent	Independent	Independent
OpenSSL < 0.9.6d (χ_1)	1.40	2.82	2.45	N/A
OpenSSL 0.9.6d (χ_2)	4.58	4.44	2.00	N/A
Apache Up-to-date (χ_3)	-	2.39	-	3.53
$\chi_1 * \chi_3$	6.14	N/A	-	-
$\chi_2 * \chi_3$	-	N/A	-	-

Figure 20 Summary of significant predictors

9 Conclusions

We have described a longitudinal study of user response to security flaws. Our study begins with the announcement of the vulnerability and continues through the release of a worm that exploits that vulnerability. We report two primary results, one regarding the rate of user response and the second regarding the composition of the responding groups.

We first observe that administrator response is poor. Three weeks after the initial vulnerability announcement, fix deployment had leveled off and 60%

of the sample was still vulnerable at 27 days. Data collected in the three weeks after the release of the Slapper worm showed a similar pattern with a projected asymptote of 32% vulnerable.

We also identified a number of predictors for server behavior, summarized in Figure 20. In the first round of upgrading, servers running recent versions of OpenSSL and Apache were overrepresented in the responding population. In the second round of upgrading, servers running recent versions of Apache were overrepresented. Even after two rounds of upgrading, a fair number of servers running relatively recent OpenSSL distributions remained vulnerable, including several "prestige" servers. In general, large hosting service providers appear to be rather more responsive. Roughly twice as high a percentage of HSPs upgraded in the first round as compared to the population average.

Acknowledgments

The author would like to thank the ICIR Wednesday Seminar, in particular Sally Floyd, Mark Handley, Vern Paxson, and Scott Shenker, for their comments on the talk that turned into this paper. Jeff Schiller warned me about the bug prior to its release, allowing me to get my probes in place. Kevin Dick, Lisa Dusseault, Allison Mankin and Terence Spies provided discussions about experimental technique and direction. Phil Beineke provided assistance with the statistical analysis. Thanks are also due to Frank Harrell and the members of the R-help mailing list for assistance with R.

References

[1] Netcraft, *Web Server Survey Home Page.* `http://www.netcraft.com/`

[2] CERT, “Apache Web Server Chunk Handling Vulnerability,” CERT Advisory CA-2002-17 (June 17, 2002). `http://www.cert.org/advisories/CA-2002-17.html`

[3] Nunes, F., “bugtraq.c httpd apache ssl attack,” Bugtraq posting (September 10, 2002.).

[4] Laurie, B., *"OpenSSL Security Alert - Remote Buffer Overflows"*, OpenSSL Mailing List (August 2002).

[5] Akkerman, W., Debian Security Advisory DSA-136-1 (July 30, 2002).

[6] *Trustix Secure Linux Advisory 2002-0063* (July 29, 2002).

[7] *EnGarde Secure Linux Security Advisory ESA-20020730-019* (July 30, 2002).

[8] Ahlberg, D., *Gentoo Linux Security Announcement* (July 7, 2002).

[9] Slashdot, *OpenSSL Security Update.* `http://developers.slashdot.org/article.pl?sid=02/07/30/1323226&mode=thread&tid=128`

[10] *FreeBSD Security Advisory FreeBSD-SA-02:33.openssl* (July 31, 2002).

[11] *Apple Security Update 2002-08-02* (August 2, 2002).

[12] *NetBSD Security Advisory 2002-009* (September 22, 2002).

[13] Benham, M., “IE SSL Vulnerability,” Bugtraq posting (August 5, 2002).

[14] OpenPKG, OpenPKG-SA-2002.008 (July 30, 2002).

[15] Hittel, S., *Modap OpenSSL Worm Analysis.* `http://analyzer.securityfocus.com/alerts/020916-Analysis-Modap.pdf`

[16] Murray, E., *SSL Server Security Survey* (July 31, 2000). `http://www.lne.com/ericm/papers/ssl_servers.html`

[17] Weimer, F., “Remote detection of vulnerable OpenSSL versions,” Bugtraq posting (September 10, 2002).

[18] Moore, D., Shannon, C.,, and Claffy, K., “Code-Red: a case study on the spread and victims of an Internet worm,” *Internet Measurement Workshop* (2002).

[19] Kleinbaum, D., and Klein, M., *Logistic Regression: A Self-Learning Text, 2ed,* Springer-Verlag, New York (2002).

[20] Hosmer, D., and Lemeshow, S., *Applied Logistic Regression, 2ed,* Wiley, New York (2000).

[21] Ihaka, R., and Gentleman, R., “R: A Language for Data Analysis and Graphics,” *Journal of Computational and Graphical Statistics.*

[22] Harrell, F. E., *Design Library.* `http://hesweb1.med.virginia.edu/biostat/s/Design.html`

[23] Harrenstien, K., Stahl, M. K., and Feinler, E. J., “NICNAME/WHOIS,” RFC 954.

[24] Beattie, S., Arnold, S., Cowan, C., Wagle, C., Wright, C., and Shostack, A., “Timing the Application of Security Patches for Optimal Uptime,” *Proceedings of LISA XVI* (2002).

[25] Rescorla, E., *ssldump.* `http://www.rtfm.com/ssldump`

[26] Dick, K., *Personal communication.*.

[27] SonicWall, *High Availability Options for SonicWALL SSL Devices.*
`http://www.sonicwall.com/products/documentation/High_Availability_SSL.pdf`

[28] Provos, N., and Honeyman, P., "ScanSSH - Scanning the Internet for SSH Servers," *16th USENIX Systems Administration Conference (LISA)* (2001).

[29] Zalewski, M., "Remote Vulnerability in SSH daemon crc32 compensation attack detector," RAZOR Bindview Advisory CAN-2001-0144 (2001).

[30] Cheswick, W., Bellovin, S., and Rubin, A., *Firewalls and Internet Security, 2nd edition* (In press.).