The Matter of Heartbleed

*Zakir Durumeric[1], James Kasten[1], David Adrian[1], J. Alex Halderman[1], Michael Bailey[1,2]
[1] University of Michigan
[2] University of Illinois, Urbana Champaign
{zakir, jdkasten, davadria, jhalderm}@umich.edu, mdbailey@illinois.edu

*Frank Li[3], Nicholas Weaver[3,4], Johanna Amann[4], Jethro Beekman[3], Mathias Payer[3,5], Vern Paxson[3,4]
[3] EECS, University of California, Berkeley
[4] International Computer Science Institute
[5] Purdue University
{frankli, nweaver, jbeekman, vern}@cs.berkeley.edu, johanna@icir.org, mpayer@purdue.edu

ABSTRACT

The Heartbleed vulnerability took the Internet by surprise in April 2014. The vulnerability, one of the most consequential since the advent of the commercial Internet, allowed attackers to remotely read protected memory from an estimated 24–55% of popular HTTPS sites. In this work, we perform a comprehensive, measurement-based analysis of the vulnerability's impact, including (1) tracking the vulnerable population, (2) monitoring patching behavior over time, (3) assessing the impact on the HTTPS certificate ecosystem, and (4) exposing real attacks that attempted to exploit the bug. Furthermore, we conduct a large-scale vulnerability notification experiment involving 150,000 hosts and observe a nearly 50% increase in patching by notified hosts. Drawing upon these analyses, we discuss what went well and what went poorly, in an effort to understand how the technical community can respond more effectively to such events in the future.

1. INTRODUCTION

In March 2014, researchers found a catastrophic vulnerability in OpenSSL, the cryptographic library used to secure connections in popular server products including Apache and Nginx. While OpenSSL has had several notable security issues during its 16 year history, this flaw—the *Heartbleed* vulnerability—was one of the most impactful. Heartbleed allows attackers to read sensitive memory from vulnerable servers, potentially including cryptographic keys, login credentials, and other private data. Exacerbating its severity, the bug is simple to understand and exploit.

In this work, we analyze the impact of the vulnerability and track the server operator community's responses. Using extensive active scanning, we assess who was vulnerable, characterizing Heartbleed's scope across popular HTTPS websites and the full IPv4 address space. We also survey the range of protocols and server products affected. We estimate that 24–55% of HTTPS servers in the Alexa Top 1 Million were initially vulnerable, including 44 of the Alexa Top 100. Two days after disclosure, we observed that 11% of HTTPS sites in the Alexa Top 1 Million remained vulnerable, as did 6% of all HTTPS servers in the public IPv4 address space. We find that vulnerable hosts were not randomly distributed, with more than 50% located in only 10 ASes that do not reflect the ASes with the most HTTPS hosts. In our scans of the IPv4 address space, we identify over 70 models of vulnerable embedded devices and software packages. We also observe that both SMTP+TLS and Tor were heavily affected; more than half of all Tor nodes were vulnerable in the days following disclosure.

Our investigation of the operator community's response finds that within the first 24 hours, all but 5 of the Alexa Top 100 sites were patched, and within 48 hours, all of the vulnerable hosts in the top 500 were patched. While popular sites responded quickly, we observe that patching plateaued after about two weeks, and 3% of HTTPS sites in the Alexa Top 1 Million remained vulnerable almost two months after disclosure.

In addition to patching, many sites replaced their TLS certificates due to the possibility that the private keys could have been leaked. We analyze certificate replacement and find that while many of the most popular websites reacted quickly, less than a quarter of Alexa Top 1 Million sites replaced certificates in the week following disclosure. Even more worryingly, only 10% of the sites that were vulnerable 48 hours after disclosure replaced their certificates within the next month, and of those that did, 14% neglected to change the private key, gaining no protection from certificate replacement.

We also investigate widespread attempts to exploit Heartbleed, as seen in extensive bulk traffic traces recorded at four sites. We find no evidence of exploitation prior to the vulnerability's public disclosure, but we detect subsequent exploit attempts from almost 700 sources, beginning less than 24 hours after disclosure. Comparing attack attempts across sites, we observe that despite the large number of sources and scans, only a handful appear to reflect exhaustive Internet-wide scans.

Finally, starting three weeks after disclosure, we undertook a large-scale notification effort and contacted the operators responsible for the remaining vulnerable servers. By contacting the operators in two waves, we could conduct a controlled experiment and measure the impact of notification on patching. We report the effects of our notifications, observing a surprisingly high 47% increase in patching by notified operators.

We draw upon these observations to discuss both what went well and what went poorly in the aftermath of Heartbleed. By better understanding the lessons of this security disaster, the technical community can respond more effectively to such events in the future.

*These authors contributed equally to this work.

IMC'14, November 5–7, 2014, Vancouver, BC, Canada.
ACM 978-1-4503-3213-2/14/11.
http://dx.doi.org/10.1145/2663716.2663755 .

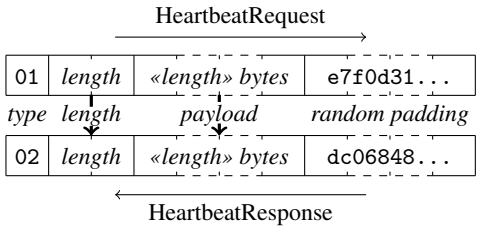

Figure 1: **Heartbeat Protocol.** Heartbeat requests include user data and random padding. The receiving peer responds by echoing back the data in the initial request along with its own padding.

2. BACKGROUND

On April 7, 2014, the OpenSSL project publicly disclosed the Heartbleed vulnerability, a bug in their implementation of the TLS Heartbeat Extension. The vulnerability allowed attackers to remotely dump protected memory—including data passed over the secure channel and private cryptographic keys—from both clients and servers. In this section, we provide a brief history of OpenSSL, the Heartbeat Extension, and details of the vulnerability and its disclosure.

2.1 OpenSSL: A Brief History

OpenSSL is a popular open-source cryptographic library that implements the SSL and TLS protocols. It is widely used by server software to facilitate secure connections for web, email, VPN, and messaging services. The project started in 1998 and began tracking vulnerabilities in April 2001.

Over the last 13 years, OpenSSL has documented six code execution vulnerabilities that allowed attackers to compromise private server data (e.g., private cryptographic keys and messages in memory) and execute arbitrary code. The project has faced eight information leak vulnerabilities, four of which allowed attackers to retrieve plaintext, and two of which exposed private keys. Two of the vulnerabilities arose due to protocol weaknesses; the remainder came from implementation errors.

The Heartbleed bug reflects one of the most impactful vulnerabilities during OpenSSL's history for several reasons: (1) it allowed attackers to retrieve private cryptographic keys and private user data, (2) it was easy to exploit, and (3) HTTPS and other TLS services have become increasingly popular, resulting in more affected services.

2.2 TLS Heartbeat Extension

The Heartbeat Extension allows either end-point of a TLS connection to detect whether its peer is still present, and was motivated by the need for session management in Datagram TLS (DTLS). Standard implementations of TLS do not require the extension as they can rely on TCP for equivalent session management.

Peers indicate support for the extension during the initial TLS handshake. Following negotiation, either end-point can send a `HeartbeatRequest` message to verify connectivity. The extension was introduced in February 2012 in RFC 6520 [66], added to OpenSSL on December 31, 2011, and released in OpenSSL Version 1.0.1 on March 14, 2012.

`HeartbeatRequest` messages consist of a one-byte type field, a two-byte payload length field, a payload, and at least 16 bytes of random padding. Upon receipt of the request, the receiving end-point responds with a similar `HeartbeatResponse` message, in which it echoes back the `HeartbeatRequest` payload and its own random padding, per Figure 1.

Date	Event
03/21	Neel Mehta of Google discovers Heartbleed
03/21	Google patches OpenSSL on their servers
03/31	CloudFlare is privately notified and patches
04/01	Google notifies the OpenSSL core team
04/02	Codenomicon independently discovers Heartbleed
04/03	Codenomicon informs NCSC-FI
04/04	Akamai is privately notified and patches
04/05	Codenomicon purchases the `heartbleed.com` domain
04/06	OpenSSL notifies several Linux distributions
04/07	NCSC-FI notifies OpenSSL core team
04/07	OpenSSL releases version 1.0.1g and a security advisory
04/07	CloudFlare and Codenomicon disclose on Twitter
04/08	Al-Bassam scans the Alexa Top 10,000
04/09	University of Michigan begins scanning

Table 1: **Timeline of Events in March and April 2014.** The discovery of Heartbleed was originally kept private by Google as part of responsible disclosure efforts. News of the bug spread privately among inner tech circles. However, after Codenomicon independently discovered the bug and began separate disclosure processes, the news rapidly became public [36, 53].

2.3 Heartbleed Vulnerability

The OpenSSL implementation of the Heartbeat Extension contained a vulnerability that allowed either end-point to read data following the payload message in its peer's memory by specifying a payload length larger than the amount of data in the `HeartbeatRequest` message. Because the payload length field is two bytes, the peer responds with up to 2^{16} bytes (~64 KB) of memory. The bug itself is simple: the peer trusts the attacker-specified length of an attacker-controlled message.

The OpenSSL patch adds a bounds check that discards the `HeartbeatRequest` message if the payload length field exceeds the length of the payload. However, while the bug is easy to conceptualize and the fix is straight-forward, the potential impact of the bug is severe: it allows an attacker to read private memory, potentially including information transferred over the secure channel and cryptographic secrets [31, 59, 67].

2.4 Heartbleed Timeline

The Heartbleed vulnerability was originally found by Neel Mehta, a Google computer security employee, in March 2014 [36]. Upon finding the bug and patching its servers, Google notified the core OpenSSL team on April 1. Independently, a security consulting firm, Codenomicon, found the vulnerability on April 2, and reported it to National Cyber Security Centre Finland (NCSC-FI). After receiving notification that two groups independently discovered the vulnerability, the OpenSSL core team decided to release a patched version.

The public disclosure of Heartbleed started on April 7, 2014 at 17:49 UTC with the version 1.0.1g release announcement [53], followed by the public security advisory [52] released at 20:37 UTC; both announcements were sent to the OpenSSL mailing list. Several parties knew of the vulnerability in advance, including CloudFlare, Akamai and Facebook. Red Hat, SuSE, Debian, FreeBSD and ALT Linux were notified less than 24 hours before the public disclosure [36]. Others, such as Ubuntu, Gentoo, Chromium, Cisco, and Juniper were not aware of the bug prior to its public release. We present a timeline of events in Table 1.

3. THE IMPACT OF HEARTBLEED

Heartbleed had the potential to affect any service that used OpenSSL to facilitate TLS connections, including popular web, mail, messaging, and database servers (Table 2). To track its damage, we performed regular vulnerability scans against the Alexa Top 1 Million domains [1] and against 1% samples of the public, non-reserved IPv4 address space. We generated these samples using random selection with removal, per ZMap's existing randomization function [30]. We excluded hosts and networks that previously requested removal from our daily HTTPS scans [29]. In this section, we analyze the impact on those services—particularly HTTPS. We have publicly released all of the data used for this analysis at https://scans.io/study/umich-heartbleed.

3.1 Scanning Methodology

We tested for the Heartbleed bug by modifying ZMap [30] to send Heartbeat requests with no payload nor padding, and the length field set to zero. Per the RFC [66], these requests should be rejected. However, vulnerable versions of OpenSSL send a response containing only padding, rather than simply drop the request. The patched version of OpenSSL—as well as other popular libraries, including GnuTLS, NSS, Bouncy Castle, PolarSSL, CyaSSL and MatrixSSL—correctly discard the request (or do not support the Heartbeat Extension).

We emphasize that this approach does not exploit the vulnerability or access any private memory—only random padding is sent back by the server. While it was later found that Heartbleed scanning caused HP Integrated Lights-Out (iLO) devices to crash [11], we received no reports of our scans disrupting these devices—likely because our approach did not exploit the vulnerability. We have publicly released our scanner at https://zmap.io.

3.2 False Negatives

Our Heartbleed scanner contained a bug that caused vulnerable sites to sometimes appear safe due to a timeout when probing individual hosts. The root cause was that the scanner labelled each host's vulnerability as false by default, rather than null or unknown. If a Heartbleed test timed out, the scanner returned the host's vulnerability status as the default false, providing no indication of a failed test. The result is a potential false negative, where the scan reports the system as immune. Note that our scanner does not however err when reporting a system as vulnerable. As we develop in this section, the likelihood of a given scan exhibiting such as false negative fortunately does not appear to depend on the particular address being scanned, and this allows us to estimate the false negative rate.

We first assessed whether some addresses were more prone to manifest false negatives than others. To do so, we compared three complete IPv4 scans and examined systems reported as vulnerable in one scan but immune in previous scans. Since the scanner does not err in reporting a host as vulnerable, any prior report of immunity reflects a false negative (assuming no one unpatches systems). We found the IP addresses associated with such false negatives spread evenly across the address space, without any apparent correlations. This observation leads us to believe the false negatives manifest in an address-independent manner.

Although our initial scanner did not fetch the web page itself, a subsequent change in the comprehensive scan (but not the incremental scans) attempted to fetch the server's home page. As the home page fetch occurred after the Heartbleed check, any reported home page data implicitly indicates that the Heartbleed test successfully completed without a timeout.

To investigate the false negative rate, we use two full scans of the IPv4 address space, one with and one without home page fetching. The full scan conducted on April 24 did not grab server home pages, while the May 1 scan did, hence we know the validity of scan results from the May 1 scan. To soundly conduct this comparison we need to remove servers that may have switched IP addresses between the two scans. To do so, we only considered servers that presented identical TLS certificates between the two scans. While this restriction potentially introduces a bias because some sites will have both patched and changed their TLS certificates, the address-independent nature of the false negatives should cause this effect to even out.

Our scanner failed to grab the server home page for 24% of the hosts in the May scan. Of these 24% of hosts, we observe 44% appear immune. False negatives could only have occurred when testing these hosts. The remaining 56% of hosts appeared vulnerable (and hence are correctly labelled). From this, we conclude that at most $(0.24 \cdot 0.44) = 0.105$, or 10.5%, of hosts were incorrectly labelled in the May 1 scan.

For the April scan, the only observable signal of a false negative is if a host was reported immune and then reported vulnerable in the May scan. We find 6.5% of hosts exhibit this behavior. Assuming that people do not unpatch their systems, this provides an estimated lower bound of 6.5% for the April scan false negative rate. This estimate represents a lower bound because we cannot determine the vulnerability in April of a host that appears immune in both scans. In that case, a false negative is a host vulnerable in April but reported as immune, and patched by May. However, we do observe that of hosts reported as vulnerable in the April scan and successfully tested in May (so the server page was retrieved), only 0.36% appeared immune in May, indicating a very modest overall patching rate between the two scans (which accords with Figure 3 below). Given that our false negatives are address-independent, we expect a similarly low patch rate for all vulnerable April hosts. Thus, while a 6.5% false negative rate is a lower bound for the April scan, the true rate should not be significantly higher.

Given the similarity of these two false negative estimates using two different scans, ultimately we conclude that the scanner exhibited a false negative rate between 6.5% and 10.5%, but that these manifest independently of the particular server scanned. Due to this address-independent behavior, we can assume a similar false negative rate for sampled scans. We attempt to account for this error whenever possible. In particular, the bias implies that any population-based survey based on a single scan underestimates the vulnerable population. Finally, for our assessment of the impact of notifications (Section 7), we only consider a given server as non-vulnerable when it consistently reports as immune in repeated scans, which would require multiple (presumably independent) false negatives to occur before introducing a bias.

3.3 Impact on Popular Websites

Determining which websites were initially vulnerable poses significant difficulties. Little attention was paid to the Heartbeat Extension prior to the vulnerability announcement, and many popular sites patched the vulnerability within hours of the disclosure. Codenomicon, one of the groups that discovered Heartbleed, speculated that 66% of HTTPS sites were vulnerable [45]. However, this number represented the Apache and Nginx market share and may well reflect an overestimate, because some operators may have disabled the extension, deployed dedicated SSL endpoints, or used older, non-vulnerable versions of OpenSSL.

Web Servers		Mail Servers		Database Servers		XMPP Servers		Other Servers	
Apache (mod_ssl) [45]	Yes	Sendmail [62]	Yes	MySQL [62]	Yes	OpenFire [12]	No	OpenVPN [54]	Yes
Microsoft IIS [46]	No	Postfix [62]	Yes	PostgreSQL [62]	Yes	Ejabberd [5]	Yes	OpenLDAP [63]	Yes
Nginx [14]	Yes	Qmail [62]	Yes	SQL Server [46]	No	Jabberd14 [70]	Yes	Stunnel [65]	Yes
Lighttpd [62]	Yes	Exim [35]	Yes	Oracle [55]	No	Jabberd2 [41]	Yes	Openswan [49]	Yes
Tomcat [17]	Yes	Courier [37]	Yes	IBM DB2 [38]	No			Telnetd-ssl [4]	Yes
Google GWS [50]	Yes	Exchange [46]	No	MongoDB [47]	Yes			OpenDKIM [3]	Yes
LiteSpeed [42]	Yes	Dovecot [35]	Yes	CouchDB [32]	No			Proftpd [64]	Yes
IBM Web Server [38]	Yes	Cyrus [48]	Yes	Cassandra [6]	No			Bitcoin Client [24]	Yes
Tengine [13]	Yes	Zimbra [56]	Yes	Redis [35]	No				
Jetty [51]	No								

Table 2: **Vulnerable Server Products.** We survey which server products were affected by Heartbleed.

3.3.1 Top 100 Websites

All of the Alexa Top 100 websites were patched within 48 hours of disclosure—prior to the start of our scans. To document the impact on these websites, we aggregated press releases, other's targeted scans, and quotes provided to Mashable, a news site that hosted one of the popular lists of sites for which users should change their passwords due to possible exposure via Heartbleed [10].

Al-Bassam completed a vulnerability scan of the Alexa Top 10,000 domains on April 8, 2014 at 16:00 UTC (22 hours after the vulnerability disclosure) [20]. His scan found 630 vulnerable sites, 3,687 supporting HTTPS but not vulnerable, and 5,683 not supporting HTTPS. Several prominent sites, including Yahoo, Imgur, Stack Overflow, Flickr, Sogou, OkCupid, and DuckDuckGo, were found vulnerable. We investigated other sites in the Alexa Top 100 and found that half made a public statement regarding vulnerability or provided information to Mashable [8, 10, 18–21, 23, 26, 28, 33, 39, 40, 50, 58, 61, 68, 69, 71, 73, 74]. Combining these press releases, Mashable's report, and Al-Bassam's scan, we find that at least 44 of the Alexa Top 100 websites were vulnerable. However, this figure reflects a lower bound, as we were unable to find information for some sites. Table 3 lists the vulnerability status of the top 30 HTTPS-enabled sites in the US.

Site	Vuln.	Site	Vuln.	Site	Vuln.
Google	Yes	Bing	No	Wordpress	Yes
Facebook	No	Pinterest	Yes	Huff. Post	?
Youtube	Yes	Blogspot	Yes	ESPN	?
Yahoo	Yes	Go.com	?	Reddit	Yes
Amazon	No	Live	No	Netflix	Yes
Wikipedia	Yes	CNN	?	MSN.com	No
LinkedIn	No	Instagram	Yes	Weather.com	?
eBay	No	Paypal	No	IMDB	No
Twitter	No	Tumblr	Yes	Apple	No
Craigslist	?	Imgur	Yes	Yelp	?

Table 3: **Vulnerability of Top 30 US HTTPS-Enabled Websites.** We aggregate published lists of vulnerable sites, press releases, and news sources to determine which of the top sites were vulnerable before the discovery of Heartbleed.

3.3.2 Estimating Broader Impact

Within 48 hours of the initial disclosure, we conducted our first vulnerability scan of the Alexa Top 1 Million. At that point, we found that 45% of all sites supported HTTPS. 60% of those supported the Heartbeat Extension, and 11% of all HTTPS sites were vulnerable. While 60% of HTTPS sites supported the extension, 91% of these were powered by known vulnerable web servers (e.g., Nginx or Apache Web Server), as shown in Table 4. If all of these servers were initially vulnerable and operators installed a patched OpenSSL version (rather than rebuilding OpenSSL with Heartbeat disabled), at most about 55% of the HTTPS sites in the Alexa Top 1 Million were initially vulnerable.

While disabling the largely unused extension would appear to provide an obvious solution, it is not possible to disable the extension through a configuration file. Instead, this change requires recompiling OpenSSL with a specific flag—an option likely more laborious than updating the OpenSSL software package.

Some sites may possibly have used an older version of OpenSSL that was not vulnerable. To estimate a lower bound for the number of vulnerable sites, we considered sites that used vulnerable web servers and supported TLS 1.1 and 1.2—features first introduced in OpenSSL 1.0.1 along with the Heartbeat Extension. Such sites would have been vulnerable unless administrators had recompiled OpenSSL to explicitly disable the extension.

To estimate the number of sites that supported TLS 1.1 and 1.2 prior to the Heartbleed disclosure, we analyzed the data collected by the Trustworthy Internet Movement's SSL Pulse [16], which provides monthly statistics of SSL-enabled websites within the Alexa Top 1 Million. We find that 56,019 of the 171,608 (32.6%) sites in the SSL Pulse dataset supported TLS 1.1 or 1.2. Of these sites, 72.7% used known vulnerable web servers, yielding an estimated lower bound of 23.7% of the sites being vulnerable.

In summary, we can reasonably bound the proportion of vulnerable Alexa Top 1 Million HTTPS-enabled websites as lying between 24–55% at the time of the Heartbleed disclosure.

Web Server	Alexa Sites	Heartbeat Ext.	Vulnerable
Apache	451,270 (47.3%)	95,217 (58.4%)	28,548 (64.4%)
Nginx	182,379 (19.1%)	46,450 (28.5%)	11,185 (25.2%)
Microsoft IIS	96,259 (10.1%)	637 (0.4%)	195 (0.4%)
Litespeed	17,597 (1.8%)	6,838 (4.2%)	1,601 (3.6%)
Other	76,817 (8.1%)	5,383 (3.3%)	962 (2.2%)
Unknown	129,006 (13.5%)	8,545 (5.2%)	1,833 (4.1%)

Table 4: **Alexa Top 1 Million Web Servers.** We classify the web servers used by the Alexa Top 1 Million Sites, as observed in our first scan on April 9, 2014. Percentages represent the breakdown of server products for each category. We note that while Microsoft IIS was not vulnerable to Heartbleed, a small number of IIS servers used vulnerable SSL terminators.

3.4 Pre-Disclosure Patching

Google, Akamai, and other sites disabled the Heartbeat Extension prior to public disclosure. To detect when services disabled the Heartbeat Extension, we examined data from the ICSI Certificate Notary, which passively monitors TLS connections from

seven research and university networks (approximately 314K active users) [22].

The Notary data shows that Google disabled Heartbeat starting at least 12 days prior to public disclosure, with all servers Heartbeat-disabled by April 15. While some servers still had Heartbeat enabled after disclosure, they may not have been exploitable. Google may have already patched those servers, and decided afterwards to disable the Heartbeat Extension as a company-wide policy. Similarly, Akamai began disabling Heartbeat at least 4 days prior to disclosure, completing the process by April 18.

3.5 Internet-Wide HTTPS Vulnerability

We began performing daily 1% scans of the IPv4 address space on April 9, 48 hours after the disclosure. Our first scan found that 11.4% of HTTPS hosts supported the Heartbeat Extension and 5.9% of all HTTPS hosts were vulnerable. Combining these proportions from random sampling with our daily scans of the HTTPS ecosystem [29] (which do not include Heartbleed vulnerability testing), we estimate that 2.0 million HTTPS hosts were vulnerable two days after disclosure.

Surprisingly, 10 ASes accounted for over 50% of vulnerable HTTPS hosts but represented only 8.6% of all HTTPS hosts (Figure 2). With the exception of Comcast Cable Communications, the ASes all belonged to web hosting companies or cloud providers (Table 5). The vulnerable hosts in the Comcast AS were Fortinet devices. In the case of Strato Hosting, vulnerable addresses were hosting Parallels Plesk Panel, a web hosting management software. The vulnerable addresses of Minotavar Computers, ZeXoTeK IT-Services, Euclid systems, Vivid Hosting, and ACCESSPEOPLE-DE all served the default Apache page, likely reflecting named-based virtual hosts. In the case of the two Amazon ASes and Hetzner Online, a large number of the vulnerable hosts served public facing websites, and used Apache or Nginx.

AS	% of Vulnerable	% of HTTPS
Minotavar Computers EOOD	18.5%	1.7%
ZeXoTeK IT-Services GmbH	13.0%	0.9%
ACCESSPEOPLE-DE ISP-Service	7.4%	0.7%
Amazon.com, Inc.	4.6%	0.8%
Amazon.com, Inc.	4.1%	0.9%
Hetzner Online AG	2.6%	0.4%
Comcast Cable Communications	2.3%	2.8%
Vivid Hosting	2.0%	0.1%
Euclid Systems	1.5%	0.1%
Strato Hosting	1.4%	0.1%
Total	57.4%	8.6%

Table 5: **Top ASes with Most Vulnerable Hosts.** We aggregate hosts by AS and find that 57% of vulnerable hosts in the April 9 scan were located in only 10 ASes.

3.6 Vulnerable Devices and Products

Heartbleed also affected many embedded systems, including printers, firewalls, VPN endpoints, NAS devices, video conferencing systems, and security cameras. To understand the embedded systems that were affected, we analyzed the self-signed certificates employed by vulnerable hosts. We clustered these by fields in the certificate Subject and manually inspected large clusters. From this, we developed device "fingerprints". We took a conservative approach and chose the most restrictive fingerprints in order to minimize false positive identifications. This, and the manual effort required, means that our fingerprints lack comprehensive coverage. However, we

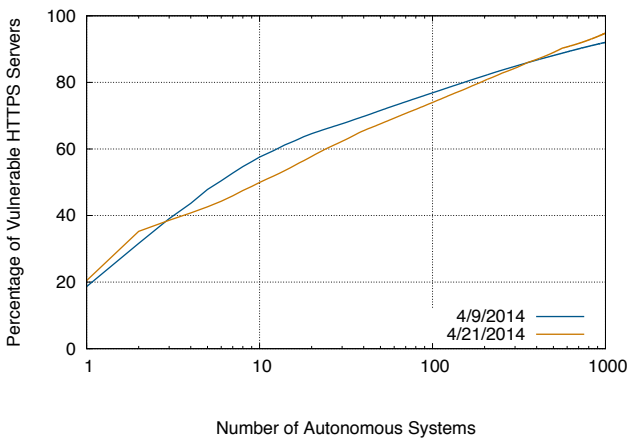

Figure 2: **Vulnerable Servers by AS.** We aggregate vulnerable hosts by AS and find that over 50% of vulnerable hosts are located in ten ASes.

still identified 74 distinct sets of vulnerable devices and software packages that fall into a number of broad categories:

Communication Servers: IceWarp messaging, Zimbra collaboration servers, iPECS VoIP systems, and Polycom and Cisco video conference products.

Software Control Panels: Puppet Enterprise Dashboard, IBM System X Integrated Management Modules, Kloxo Web hosting control panel, PowerMTA, Chef/Opscode management consoles, VMWare servers, and Parallels control panels for Plesk and Confixx.

Network Attached Storage: QNAP, D-Link, ReadyNAS, LaCie, Synology, and Western Digital NAS devices.

Firewall and VPN Devices: Devices from Barracuda Networks, Cisco, SonicWALL, WatchGuard, OpenVPN, pfSense, TOPSEC Network Security (a Chinese vendor), and Fortinet.

Printers: Dell, Lexmark, Brother, and HP printers.

Miscellaneous: Hikvision and SWANN security cameras, AcquiSuite power monitors, IPACCT (a management tool used by Russian ISPs), Aruba Networks WiFi login portals, INSYS VPN access for industrial controllers, and SpeedLine Solutions (the "#1-rated Pizza POS System").

3.7 Other Impacts

While our study focuses on Heartbleed's impact on public HTTPS web services, Heartbleed also affected mail servers, the Tor network, Bitcoin, Android, and wireless networks, as we briefly assess in this section.

Mail Servers. SMTP, IMAP, and POP3 can use TLS for transport security via use of a `StartTLS` directive within a plaintext session. As such, if mail servers used OpenSSL to facilitate TLS connections, they could have been similarly vulnerable to Heartbleed. On April 10, we scanned a random 1% sample of the IPv4 address space for vulnerable SMTP servers. We found that 45% of those providing SMTP+TLS supported the Heartbeat Extension, and 7.5% were vulnerable to Heartbleed.

These estimates only provide a lower bound, because similar to HTTPS, our scanner sometimes timed out, causing false negatives. (We also scanned for IMAP and POP3 servers, but later analysis of the data found systematic flaws that rendered the results unusable.)

Tor Project. Tor relays and bridges use OpenSSL to provide TLS-enabled inter-relay communication. In our April 10 scan, we found that 97% of relays supported Heartbeat and could have been

vulnerable. 48% of the relays remained vulnerable at that time, three days after announcement of the vulnerability. The vulnerability could have allowed an attacker to extract both short-term onion and long-term identity keys, ultimately allowing an attacker to intercept traffic and impersonate a relay. In the case of a hidden service, the bug could have allowed an entry guard to locate a hidden service. The Tor client was similarly affected, potentially allowing entry guards to read sensitive information from a client's memory, such as recently visited websites [27].

Bitcoin Clients. Heartbleed affected both Bitcoin clients and exchanges, and in the most severe case, allowed an attacker to compromise the accounts on a popular exchange, BTCJam [15]. The value of Bitcoin did not change drastically on April 7, the date of public disclosure, falling only 3.1% from the previous day (a figure within its regular volatility) and returned to its April 6 value by April 14.

All versions of the Bitcoin software from May 2012 to April 2014 used a vulnerable OpenSSL version [2]. Immediately after Heartbleed's disclosure, a new Bitcoin version was released linking to the newly patched OpenSSL version. Because clients were also affected by the bug, attackers could potentially compromise wallets or retrieve private keys if a susceptible user followed a payment request link [24]. However, we have not found any reports of the theft of Bitcoins from local wallets.

Several companies, including Bitstamp and Bitfinex, temporarily suspended transactions on April 8 until they could patch their servers. In the most severe case, 12 customers had a total of 28 BTC (≈ $6,500) stolen from BTCJam after account credentials were compromised, though with all funds subsequently reinstated by the exchange [15].

Android. Heartbleed only affected Android version 4.1.1 [50]. It is unclear how many devices currently run the affected version, but Google recently estimated that 33.5% of all Android devices currently run Android 4.1.x [7]. A vulnerable device would have been susceptible to having memory read by a malicious server.

Wireless Networks. Several variants of the Extended Authentication Protocol, a commonly used framework for wireless network authentication, use TLS, including EAP-PEAP, EAP-TLS, and EAP-TTLS. For implementations based on OpenSSL, Heartbleed would have allowed attackers to retrieve network keys and user credentials from wireless clients and access points [34]. We do not know of any statistics regarding what sort of vulnerable population this potentially represents.

4. PATCHING

In Section 3, we estimated the initial impact of Heartbleed. In this section, we discuss the patching behavior that occurred subsequent to the disclosure.

4.1 Popular Websites

Popular websites did well at patching. As mentioned above, only five sites in the Alexa Top 100 remained vulnerable when Al-Bassam completed his scan 22 hours after disclosure. All of the top 100 sites were patched by the time we started our scans, 48 hours after disclosure. As discussed in Section 3, our first scan of the Alexa Top 1 Million found that 11.5% of HTTPS sites remained vulnerable. The most popular site that remained vulnerable was `mpnrs.com`, ranked 689th globally and 27th in Germany. Similarly, all but seven of the vulnerable top 100 sites replaced their certificate in the first couple of weeks following disclosure. Most interestingly, `godaddy.com`, operator of the largest commercial certificate authority, did not change their certificates until much later. The other six sites are `mail.ru`, `instagram.com`, `vk.com`, `sohu.com`, `adobe.com`, and `kickass.to`.

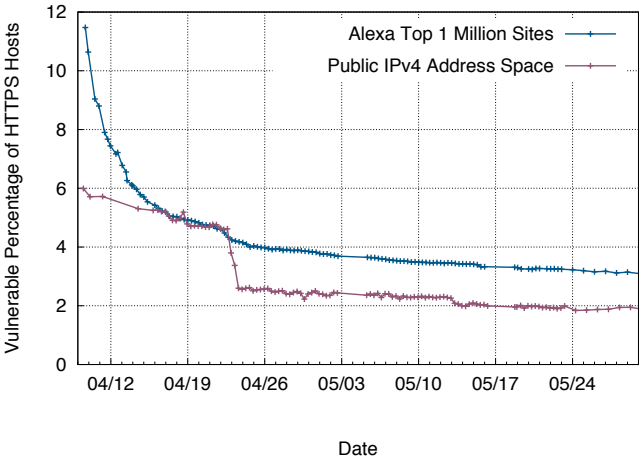

Figure 3: **HTTPS Patch Rate.** We track vulnerable web servers in the Alexa Top 1 Million and the public IPv4 address space. We track the latter by scanning independent 1% samples of the public IPv4 address space every 8 hours. Between April 9 and June 4, the vulnerable population of the Alexa Top 1 Million shrank from 11.5% to 3.1%, and for all HTTPS hosts from 6.0% to 1.9%.

As shown in Figure 3, while many Alexa Top 1 Million sites patched within the first week, the patch rate quickly dropped after two weeks, with only a very modest decline between April 26 and June 4, 2014. While top sites in North America and Europe were initially more vulnerable than Asia and Oceania (presumably due to more prevalent use of OpenSSL-based servers), they all followed the same general patching pattern visible in Figure 3.

4.2 Internet-Wide HTTPS

As can be seen in Figure 3, the patching trend for the entire IPv4 address space differs from that of the Alexa Top 1 Million. The largest discrepancy occurs between April 22, 14:35 EDT and April 23, 14:35 EDT, during which the total number of vulnerable hosts fell by nearly a factor of two, from 4.6% to 2.6%. This dropoff occurred because several heavily affected ASes patched many of their systems within a short time frame. The shift from 4.6% to 3.8% between 14:35 and 22:35 on April 22 reflects Minotavar Computers patching 29% of their systems, ZeXoTeK IT-Services patching 60%, and Euclid Systems patching 43%. Between April 22, 22:35 and April 23, 06:35, Minotavar Computers continued to patch systems. The last major drop from 3.4% to 2.6% (06:35–14:35 on April 23) was primarily due to Minotavar Computers patched remaining systems, and to a lesser extent, Euclid Systems and Vivid Hosting.

4.3 Comparison to Debian Weak Keys

In 2008, a bug was discovered in the Debian OpenSSL package, in which the generation of cryptographic keys had a severely limited source of entropy, reducing the space of possible keys to a few hundred thousand. The lack of entropy allowed attackers to fully enumerate the SSL and SSH keys generated on Debian systems, thus making it vital for Debian OpenSSL users to generate fresh replacement keys.

Yilek et al. measured the impact of the vulnerability and patching behavior by performing daily scans of HTTPS servers [72]. Given the similarities in the severity and nature of remediation between

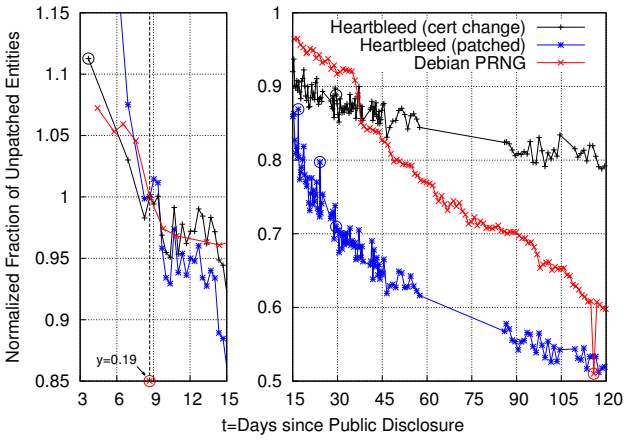

Figure 4: **Comparison of the Patch Rates of the Debian PRNG and Heartbleed Vulnerabilities.** The y-axis is normalized at 8.7 days, indicated by the vertical striped line. Thus, the fraction of unpatched entities at a given time is relative to the fraction at 8.7 days after disclosure, for each dataset. Except for the points marked by ○, for each measurement the size of the Debian PRNG entity population was $n = 41{,}200 \pm 2{,}000$, and for Heartbleed, $n = 100{,}900 \pm 7{,}500$. Due to a misconfiguration in our measurement setup, no Heartbleed data is available days 58–85.

this event and Heartbleed, we compared the community's responses to both disclosures.

A key methodological issue with conducting such a comparison concerns ensuring we use an "apples-to-apples" metric for assessing the extent of the community's response to each event. The comparison is further complicated by the fact that our Heartbleed measurements sample a different 1% of the Internet each scan. We do the comparison by framing the basic unit of "did an affected party respond" in terms of aggregate entities very likely controlled by the same party (and thus will update at the same time). To do so, we define an entity as a group of servers that all present the same certificate during a particular measurement. This has the potential for fragmenting groups that have partially replaced their certificates, but we argue that this effect is likely negligible since the number of entities stays roughly constant across our measurements. Note that this definition of entity differs from the "host-cert" unit used in [72], in which groups were tracked as a whole from the first measurement.

Figure 4 shows for both datasets the fraction of unfixed entities to the total number of entities per measurement. We consider an entity as "fixed" in the Debian PRNG dataset if the certificate now has a strong public key (and previously did not), otherwise "unfixed". For our Heartbleed IPv4 dataset (labelled "patch"), we deem an entity as "fixed" if *all* servers presenting the same certificate are now no longer vulnerable, "unfixed" otherwise.

This data shows that entities vulnerable to Heartbleed patched somewhat more quickly than in the Debian scenario, and continue to do so at a faster rate. It would appear that aspects of the disclosure and publicity of Heartbleed did indeed help with motivating patching, although the exact causes are difficult to determine.

Note that for the Debian event, it was very clear that affected sites had to not only patch but to also issue new certificates, because there was no question that the previous certificates were compromised. For Heartbleed, the latter step of issuing new certificates was not as pressing, because the previous certificates were compromised only if attackers had employed the attack against the site prior to patching *and* the attack indeed yielded the certificate's private key.

Given this distinction, we also measured whether entities changed their certificates after patching Heartbleed.* To do so, we now define an entity as a group of servers that all present the same certificate during both a particular measurement and all previous measurements. We regard an entity as "unfixed" if *any* server presenting that certificate is vulnerable at any time during this time frame and "fixed" otherwise. Again, we argue that fragmentation due to groups having their servers only been partially patched is likely negligible. We label this data as "cert change" in Figure 4. We see that while entities patched Heartbleed faster than the Debian PRNG bug, they replaced certificates more slowly, which we speculate reflects a perception that the less-definitive risk of certificate compromise led a number of entities to forgo the work that reissuing entails.

5. CERTIFICATE ECOSYSTEM

Heartbleed allowed attackers to extract private cryptographic keys [67]. As such, the security community recommended that administrators generate new cryptographic keys and revoke compromised certificates [35]. In this section, we analyze to what degree operators followed these recommendations.

5.1 Certificate Replacement

To track which sites replaced certificates and cryptographic keys, we combined data from our Heartbleed scans, Michigan's daily scans of the HTTPS ecosystem [29], and ICSI's Certificate Notary service [22]. Of the Alexa sites we found vulnerable on April 9 (2 days after disclosure), only 10.1% replaced their certificates in the month following disclosure (Figure 5). For comparison, we observed that 73% of vulnerable hosts detected on April 9 patched in that same time frame, indicating that most hosts who patched did not replace certificates. In addition, it is striking to observe that only 19% of the vulnerable sites that did replace their certificates also revoked the original certificate in the same time frame, and even more striking that 14% *re-used the same private key*, thus gaining no actual protection by the replacement.

We find that 23% of all HTTPS sites in the Alexa Top 1 Million replaced certificates and 4% revoked their certificates between April 9 and April 30, 2014. While it may seem inverted that fewer vulnerable sites changed their certificates compared to all HTTPS sites in the Alexa Top 1 Million, our first scan was two days after initial disclosure. We expect that diligent network operators both patched their systems and replaced certificates within the first 48 hours post disclosure, prior to our first scan.

The ICSI Certificate Notary (see Section 6.1) provides another perspective on changes in the certificate ecosystem, namely in terms of Heartbleed's impact on the sites that its set of users visit during their routine Internet use. In Figure 6, we show the difference in certificate replacement between March and April 2014. For the first four days after public disclosure on April 7, we observed a large drop in the number of servers with the same certificate as on April 6, indicating a spike in new certificates. After that, certificate changes progressed slowly yet steadily for the rest of the month. This matches our expectations that a number of operators patched their systems prior to our scans and immediately replaced certificates.

Ultimately, while popular websites did well at patching the actual vulnerability, a significantly smaller number replaced their

*See Section 5.1 for a discussion on the replacement of public/private key pairs in addition to certificates.

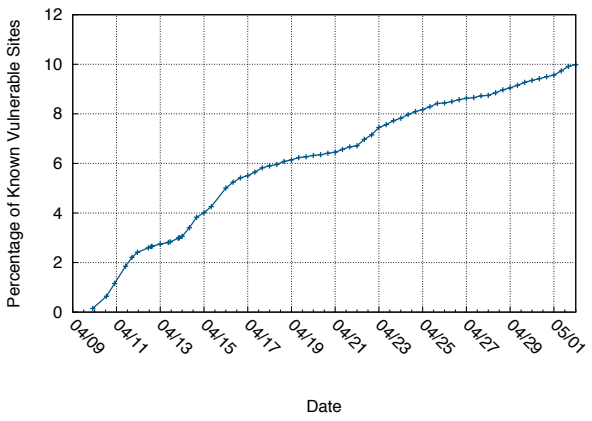

Figure 5: **Certificate Replacement on Vulnerable Alexa Sites.** We monitored certificate replacement on vulnerable Alexa Top 1 Million sites and observe only 10% replaced certificates in the month following public disclosure.

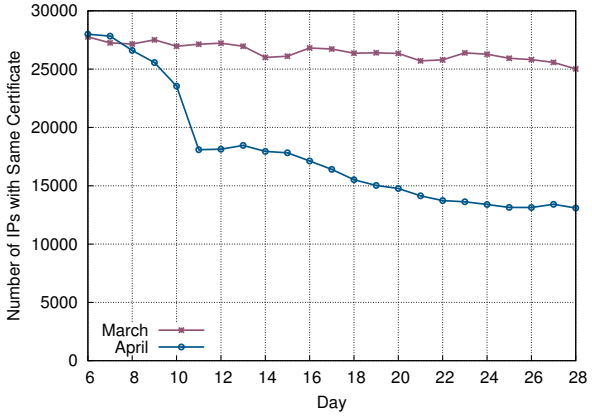

Figure 6: **ICSI Notary Certificate Changes.** Over both March and April, we track the number of servers who have the same certificate as on the 6th of each month. We only include servers that served the same certificate for the previous month.

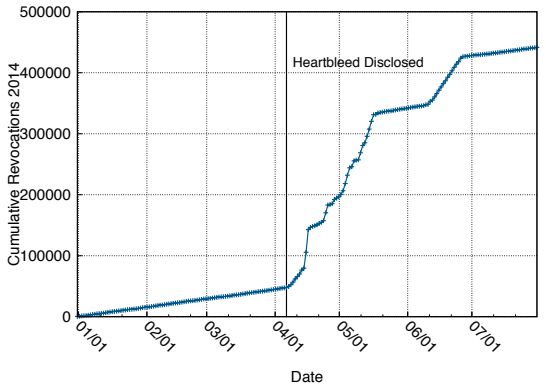

Figure 7: **Revocations after Heartbleed.** Certificate revocations dramatically increased after the disclosure. The jumps reflect GlobalSign (first) and GoDaddy (rest). However, still only 4% of HTTPS sites in the Alexa Top 1 Million revoked their certificates between April 9 and April 30, 2014.

certificates, and an even smaller number revoked their vulnerable certificates.

5.2 Certificate Revocation

When a certificate or key can no longer be trusted, sites can request the issuing CA to *revoke* the certificate. CAs accomplish this by publishing certificate revocation lists (CRLs) and supporting the Online Certificate Status Protocol (OCSP) for live queries. Even though most vulnerable hosts failed to revoke old certificates, we observed about as many revocations in the three months following public disclosure as in the three previous years.

Prior to the vulnerability disclosure, we saw on average 491 (σ=101) revocations per day for certificates found in our scans. As seen in Figure 7, in the days following disclosure, the number of revocations dramatically increased. The sudden increases were due to individual CAs invalidating large portions of their certificates. Most notably, GlobalSign revoked 56,353 certificates over two days (50.2% of their visible certificates), and GoDaddy, the largest CA, revoked 243,823 certificates in week-long bursts over the following three months. GlobalSign's large number of revocations were precipitated by a major customer, CloudFlare, revoking all of their customers' certificates [60].

Revoking such a large number of certificates burdens both clients and servers. Clients must download large CRLs, which CAs must host. GlobalSign's CRL expanded the most, from 2 KB to 4.7 MB due to CloudFlare's revocations. CloudFlare hesitated to revoke their certificates, citing its significant cost, which they estimated would require an additional 40 Gbps of sustained traffic, corresponding to approximately $400,000 per month [60]. StartCom, a CA that offers free SSL certificates, came under fire for continuing their policy of charging for revocation after the Heartbleed disclosure [44]. However, revocation places a sizable financial strain on CAs due to bandwidth costs [9, 60].

5.3 Forward Secrecy

Heartbleed highlights the importance of using *forward secret* cipher suites in TLS. Without forward secrecy, the compromise of a server's private key, such as due to Heartbleed, allows an attacker who recorded previous communications encrypted with TLS to recover the session key used to protect that communication, subverting its confidentiality. Thus, we might expect operators to respond to Heartbleed by ensuring that at least in the future their servers will support forward secrecy.

Unfortunately, we find that only 44% of the connections observed by the ICSI Notary in May 2014 used forward secrecy. There has been a slow increase in adoption between December 2013 and April 2014, a gain of 1.0–4.3% each month. We observe a 4.2% increase between March and April 2014, but no larger than that between January and February. Surprisingly, this trend stagnated from April to August; the percentage of connections using forward secrecy stayed virtually the same. Currently, we are not sure why the increase in forward secrecy cipher use ceased. However, it appears clear that Heartbleed did not spur adoption of forward secrecy.

6. ATTACK SCENE

In addition to tracking vulnerable servers, we analyzed who was scanning for the Heartbleed vulnerability by examining network traffic collected from passive taps at Lawrence Berkeley National Laboratory (LBNL), the International Computer Science Institute (ICSI), and the National Energy Research Scientific Computing Center (NERSC), as well as a honeypot operated by a colleague on Amazon EC2.

To detect Heartbleed scanning, we extended the Bro's SSL/TLS analyzer to recognize Heartbeat messages [25, 57]. Note that this approach parses the full TLS protocol data stream, including the TLS record layer which remains unencrypted throughout the session, and thus achieves an accuracy significantly better than that provided by simple byte pattern matching. We have released our Bro modifications along with our detection script via the Bro git repository.

6.1 Pre-Disclosure Activity

LBNL's network spans two /16s, one /20 and one /21. The institute frequently retains extensive packet traces for forensic purposes, and for our purposes had full traces available from February–March 2012, February–March 2013, and January 23–April 30 2014. ICSI uses a /23 network, for which we had access to 30-days of full traces from April 2014. NERSC has a /16 network, for which we analyzed traces from February to April 2014. The EC2 honeypot provided full packet traces starting in November 2013.

For all four networks, over these time periods our detector found no evidence of any exploit attempt up through April 7, 2014. This provides strong evidence that at least for those time periods, no attacker with prior knowledge of Heartbleed conducted widespread scanning looking for vulnerable servers. Such scanning however could have occurred during other time periods.

6.2 Post-disclosure Activity

To detect post-disclosure scanning, we similarly examined packet traces from LBNL, ICSI, and the EC2 honeypot. The first activity we observed originated from a host at the University of Latvia on April 8, starting at 15:18 UTC (21 hours 29 minutes after public disclosure), targeting 13 hosts at LBNL. This first attack was unusual in that it sent both unencrypted (pre-handshake) and encrypted (post-handshake) Heartbleed exploit packets to each host, likely trying to gauge the effectiveness of both approaches. We observed scanning of the other two networks within a few more hours.

In total, we observed 5,948 attempts to exploit the vulnerability from 692 distinct hosts (Table 7). These connections targeted a total 217 hosts. 7 attackers successfully completed 103 exploit attempts against 12 distinct hosts (excluding the intentionally vulnerable honeypot). Figure 8 presents the temporal behavior of scanning as seen at each location.

We detected several different types of exploit attempts, which we list in Table 6. In types (1) and (2), attackers sent exploit attempts prior to completely establishing the TLS session, which allowed us to directly inspect the attack payload. After establishment of the TLS session, the only pieces of information we can retrieve are the message type and size, which are both transferred in the clear. To detect scanning conducted in a fashion similar to our own, we checked the length of the encrypted message (3). We also consider all Heartbeat messages that we see prior to the first transmission of application data (4) to reflect exploit attempts.* We identify attacks as successful when the destination server responds with more data than included in the original request (5), though as noted above we do not consider attacks on the EC2 honeypot as "successful".

The sixteen most aggressive probe sources (by number of scans) reside in Amazon address space. The scans originating from these hosts share many characteristics, including the use of encryption, the same packet length, and identical cipher suites. Closer examination reveals that these hosts belong to two popular Heartbleed scan services: `filippio.io` (3,964 attempts from 40 hosts) and `ssllabs.com` (16 attempts from 5 hosts).

*Heartbeat messages should never precede application data. We verified this does not happen in real-world traffic by manually reviewing traces prior to the Heartbleed disclosure. We observed no such instances.

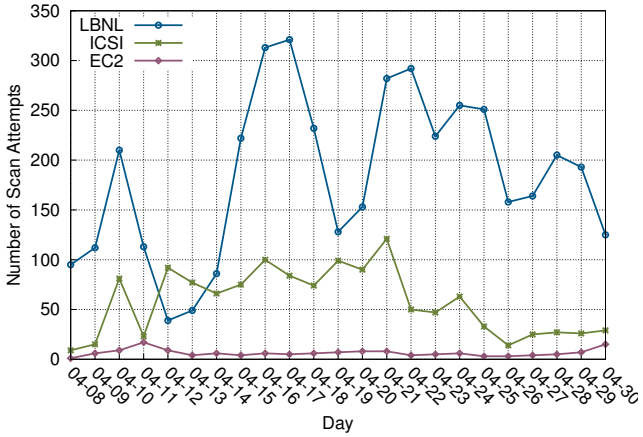

Figure 8: **Incoming Attacks.** We track the number of incoming connections containing Heartbleed exploit messages seen at ICSI, LBNL, and the EC2 honeypot per day.

Type	Sources	Targets	Connections
(1) Cleartext attacks	628	191	1,691
(2) Successful cleartext attacks	4	10	26
(3) Short Heartbeats	8	10	187
(4) Heartbeats before data	61	132	4,070
(5) Successful attacks after encryption	5	6	77
Total	692	217	5,948

Table 6: **Attack Types.** We list different stages of attacks observed against LBNL, ICSI and the EC2 honeypot.

Examining connection attempts across each site, we find only three sources that attempted to scan all three networks. However, LBNL aggressively blocks scan traffic, so most hosts scanning it were likely blocked before they could locate a server against which to attempt Heartbleed exploits. Considering only the EC2 honeypot and the ICSI network (neither of which block scanning), we find 11 sources that scanned both.

Apart from our scans at the University of Michigan and another research group at TU Berlin, we found four sources that targeted more than 100 addresses at ICSI. Two of the sources were located in CHINANET, one at Nagravision, and one at Rackspace. The remaining Heartbleed exploit attempts only targeted smaller numbers of hosts without performing widespread scanning.

Hosts performing widespread scans exclusively targeted port 443 (HTTPS). We observed a small number of exploit against ports 993 (IMAPS), 995 (POP3S), 465 (SMTPS), as well as GridFTP. We also found a small number of exploit attempts against services running on other ports.

While we observed Heartbleed attacks originating from a large number of sources, we find that most hosts did not target more than one of our sites and likely do not represent Internet-wide scanning. Given the low volume of widespread scanning, the 201 sources attempting to exploit the EC2 honeypot appears surprisingly high. Our hypothesis is that attackers may preferentially scan denser address spaces, such as those of Amazon EC2, as they will likely yield a greater number of vulnerable targets.

AS Name	ASN	Scans	Hosts
Amazon.com	14618	4,267	206
China Telecom	4812	507	139
China169 Backbone	4837	147	34
Chinanet	4134	115	23
University of Michigan	36375	92	3
SoftLayer	36351	85	2
University of Latvia	8605	50	1
Rackspace	19994	47	11
GoDaddy.com	26496	34	15
OVH	16276	30	9
ViaWest	13649	30	1
Guangdong Mobile	9808	29	1
New York Internet Company	11403	29	1
ServerStack	46652	27	1
Comcast	7922	20	5
Global Village Telecom	18881	19	4
China Telecom	4816	16	1
Turk Telekomunikasyon	9121	15	8
DFN	680	15	2
Amazon.com	16509	12	5
TekSavvy Solutions	5645	11	2
Oversun	48172	11	2
HKNet	4645	11	1
CariNet	10439	10	8
PowerTech	5381	10	1
Nagravision	42570	10	1
CYBERDYNE	37560	10	1

Table 7: **ASes Responsible for Attacks.** We list the ASNs/ISPs that sent 10 or more Heartbleed exploit attempts to LBNL, ICSI, and our EC2 honeypot.

7. NOTIFICATION

Three weeks after the initial disclosure a large number of hosts remained vulnerable, and we began notifying the system operators responsible for unpatched systems. This endeavor provided us with an opportunity to study the impact of large-scale vulnerability notification. In this section we describe our notification methodology and analyze the reactions to our notifications and its impact on patching.

7.1 Methodology

In order to find the appropriate operators to contact, we performed WHOIS lookups for the IP address of each vulnerable host appearing in our April 24, 2014 scan of the full IPv4 address space. We used the "abuse" e-mail contact extracted from each WHOIS record as our point of notification. We chose to use WHOIS abuse emails because they struck us as more reliable than emails from other sources. There also appeared to be less risk in offending a network operator through contacting the abuse contact. For example, many emails extracted from certificate Subject fields were not valid emails, and we observed several WHOIS records with comments instructing anything related to spam or abuse be sent to the abuse contact rather than the technical contact.

Our scan found 588,686 vulnerable hosts. However, we excluded embedded devices—which accounted for 56% of vulnerable hosts—because administrators likely had no avenue for patching many of these devices at the time. These devices were detected using the fingerprints described in Section 3.6. The remaining 212,805 hosts corresponded to 4,648 distinct abuse contacts. Approximately 30,000 hosts belonged to RIPE and Amazon each. Because neither of these organizations directly manage hosts, we omitted them from our notifications.

To measure the impact of our notifications, we randomly split the abuse contacts into two groups, which we notified in stages.

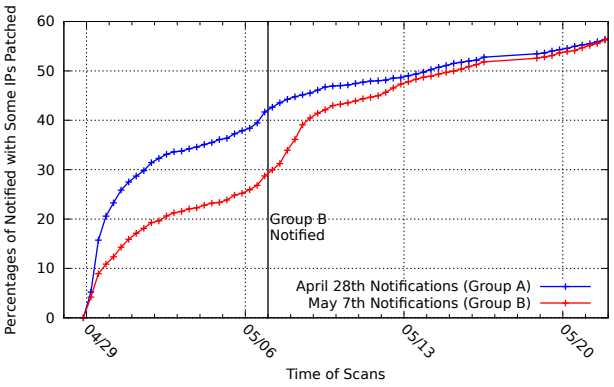

Figure 9: **Patch Rates of Group A vs Group B.** The patch rates for our two notification sets show that notification had statistically significant impact on patch rate.

We sent notifications to the first half (Group A) on April 28, 2014, and the second half (Group B) on May 7, 2014. Our notification e-mail introduced our research and provided a list of vulnerable hosts, information on the vulnerability, and a link to the EFF's Heartbleed recovery guide for systems administrators.

7.2 Patching Behavior

To track patching behavior, we conducted regular scans of the known vulnerable hosts every eight hours. We considered a contact as having reacted and begun patching if we found at least one host in the list we sent to the contact as patched. Figure 9 shows a comparison of the patch rates between the two groups. Within 24 hours of the initial notification, 20.6% of the Group A operators had begun to patch, whereas only 10.8% of Group B contacts (not yet notified) had reacted. After eight days (just before the second group of notifications), 39.5% of Group A contacts had patched versus 26.8% in Group B. This is a 47% increase in patching for notified operators.

Fisher's Exact Test yields a one-sided p-value of very nearly zero for the null hypothesis that both groups reflect identical population characteristics. We thus conclude that our notification efforts had a statistically significant positive effect in spurring notified sites to patch. Our second round of notifications followed a similar pattern as the first. As Group A's rate of patching had decreased at that point, Group B caught up, resulting in both converging to around 57% of contacts having reacted within a couple of weeks of notification.

We also investigated the relationship between the reactions of network operators (per Section 7.3) and their patching behavior. First, we sent our notification message in English, possibly creating a language barrier between us and the contact. We analyzed the Group A responses and found that email responses entirely in English had no statistically significant difference in the corresponding patching rate than for responses containing non-English text (Fisher's Exact Test yielded a two-sided p-value of 0.407).

We did, however, find statistically significant differences between each of the categories of responses framed below in Section 7.3, as shown in Figure 10, with human responders patching at the highest rate. Within the first day post-notification, 48% of human responders had begun patching, while none of the other categories had a patch rate higher than 32%.

The second strongest reactions came from contacts configured to send automated responses. 32% had reacted after one day, and 75% had reacted after three weeks. This indicates that operators using a

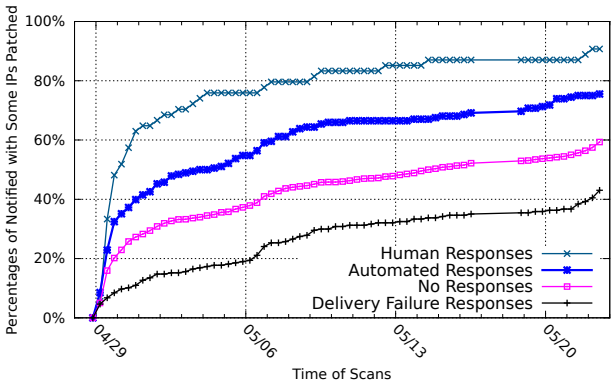

Figure 10: **Patch Rates for Different Response Types.** Conditioning on the sort of reply we received for a given notification reveals statistically significant differences.

system to automatically process notifications and complaints will still often react appropriately.

Over 77% of the contacts never responded. After one day, 20% of such contacts had conducted some patching; after three weeks, 59% had. Right before Group B's notifications, the patch rate of these contacts was statistically significantly higher than Group B's patch rate. This shows that even when system operators do not respond when notified, they often still patch vulnerable systems.

7.3 Responses

In our first group of notifications, on April 28, 2014, we contacted 2,308 abuse contacts and received email responses from 514 contacts. Of these 59 (11%) were clearly human-generated, 197 (38%) were automated, and 258 (50%) were delivery failures. We received 16 automated emails where we subsequently received a human response; these we classified as human (thus, in total we received 530 emails). The vast majority of responses (88%) were in English; other common languages included Russian, German, Portuguese, and Spanish.

We classified a positive response as one that thanked us or stated their plan to remedy their vulnerable hosts. The human-generated responses were overall very positive (54/59), with only three that we deemed neutral, and two negative. The two negative responses questioned the legality of our scan despite our explicit explanation that we did not exploit the vulnerability.

Automated messages came in four forms: confirmations (24%), tickets (44%), trackers (23%), and miscellaneous bounces (9%; primarily out-of-office notices and "no longer working here" messages). Confirmation emails confirmed the receipt of our notification; tickets provided a reference or ticket identifier to associate with our notification message; and trackers were tickets that also explicitly provided a link to a support site to track progress on opened tickets. Curiously, 21 of the 45 trackers did not provide the credentials to log into the support website, 2 provided invalid credentials, and 3 did not have our ticket listed on their support site. In the week following our notification, we were informed that 19 tickets were closed, although only 4 provided any reasoning.

Out of the 258 delivery failure replies, 197 indicated the recipient did not receive our notification. Other error messages included full inboxes or filtering due to spam, and several did not describe a clear error. We observed 30 delayed and retrying emails, but all timed-out within three days.

7.4 Network Operator Survey

We sent a brief survey to positive human responders, where all questions were optional, and received anonymous submissions from 17 contacts. Surprisingly, all 17 expressed awareness of the vulnerability and stated their organizations had performed some remediation effort prior to our notification, typically through informing their clients/customers and patching machines if accessible. When we asked why might the hosts we detected still be vulnerable, the most common responses were that they did not have direct control over those servers, or their own scans must have missed those hosts. It appears ignorance of the vulnerability and its threat did not play a factor in slow patching, although our sample size is small. When asked if they replaced or revoked vulnerable certificates, nine said yes, two said no, and one was unaware of the recommendation. Finally, we asked if these contacts would like to receive notifications of similar vulnerabilities in the future. Twelve said yes, two said no, and the others did not respond. This again demonstrates that our notifications were in general well-received.

8. DISCUSSION

Heartbleed's severe risks, widespread impact, and costly global cleanup qualify it as a security disaster. However, by analyzing such events and learning from them, the community can be better prepared to address major security failures in the future. In this section, we use our results to highlight weaknesses in the security ecosystem, suggest improved techniques for recovery, and identify important areas for future research.

HTTPS Administration. Heartbleed revealed important shortcomings in the public key infrastructure that underlies HTTPS. One set of problems concerns certificate replacement and revocation. As discussed in Section 5, only 10% of known vulnerable sites replaced their certificates, and an astounding 14% of those reused the existing, potentially leaked, private key. This data suggests that many server administrators have only a superficial understanding of how the HTTPS PKI operates or failed to understand the consequences of the Heartbleed information leak. This underscores the importance for the security community of providing specific, clear, and actionable advice for network operators if similar vulnerabilities occur in the future. Certificate management remains difficult for operators, highlighting the pressing need for solutions that enable server operators to correctly deploy HTTPS and its associated infrastructure.

One of the ironies of Heartbleed was that using HTTPS, a protocol intended to provide security and privacy, introduced vulnerabilities that were in some cases more dangerous than those of unencrypted HTTP. However, we emphasize that HTTPS is ultimately the more secure protocol for a wide variety of threat models. Unfortunately, only 45% of the Top 1 Million websites support HTTPS, despite efforts by organizations such as the EFF and Google to push for global HTTPS deployment.

Revocation and Scalability. Even though only a small fraction of vulnerable sites revoked their certificates, Heartbleed placed an unprecedented strain on certificate authorities and revocation infrastructure. In the three months following public disclosure, about as many revocations were processed by CAs as in the three years proceeding the incident. Wholesale revocation such as required by an event like Heartbleed stresses the scalability of basing revocation on the distribution of large lists of revoked certificates. As a result, CAs were backlogged with revocation processing and saddled with unexpected financial costs for CRL distribution—CloudFlare alone paid $400,000 per month in bandwidth [60]. The community needs to de-

velop methods for scalable revocation that can gracefully accommodate mass revocation events, as seen in the aftermath of Heartbleed.

Support for Critical Projects. While not a focus of our research, many in the community have argued that this event dramatically underscores shortcomings in how our community develops, deploys, and supports security software. Given the unending nature of software bugs, the Heartbleed vulnerability raises the question of why the Heartbeat extension was enabled for popular websites. The extension is intended for use in DTLS, an extension unneeded for these sites. Ultimately, the inclusion and default configuration of this largely unnecessary extension precipitated the Heartbleed catastrophe. It also appears likely that a code review would have uncovered the vulnerability. Despite the fact that OpenSSL is critical to the secure operation of the majority of websites, it receives negligible support [43]. Our community needs to determine effective support models for these core open-source projects.

Vulnerability Disclosure. With the exception of a small handful, the most prominent websites patched within 24 hours. In many ways, this represents an impressive feat. However, we also observed vulnerability scans from potential attackers within 22 hours, and it is likely that popular sites were targeted before the onset of large, indiscriminate scans.

Several factors indicate that patching was delayed because the Heartbleed disclosure process unfolded in a hasty and poorly coordinated fashion. Several major operating system vendors were not notified in advance of public disclosure, ultimately leading to delayed user recovery. As discussed in Section 3, a number of important sites remained vulnerable more than 24 hours after initial disclosure, including Yahoo, the fourth most popular site on the Internet. The security community needs to be better prepared for mass vulnerability disclosure before a similar incident happens again. This includes addressing difficult questions, such as how to determine which software maintainers and users to notify, and how to balance advance disclosure against the risk of premature leaks.

Notification and Patching. Perhaps the most interesting lesson from our study of Heartbleed is the surprising impact that direct notification of network operators can have on patching. Even with global publicity and automatic update mechanisms, Heartbleed patching plateaued two weeks after disclosure with 2.4% of HTTPS hosts remaining vulnerable, suggesting that widespread awareness of the problem is not enough to ensure patching. However, as discussed in Section 7, when we notified network operators of the unpatched systems in their address space, the rate of patching increased by 47%. Many operators reported that they had intended to patch, but that they had missed the systems we detected.

Although Internet-wide measurement techniques have enabled the mass detection of vulnerable systems, many researchers (including us) had assumed that performing mass vulnerability notifications for an incident like Heartbleed would be either too difficult or ineffective. Our findings challenge this view. Future work is needed to understand what factors influence the effectiveness of mass notifications and determine how best to perform them. For instance, was Heartbleed's infamy a precondition for the high response rate we observed? Can we develop systems that combine horizontal scanning with automatically generated notifications to quickly respond to future events? Can we standardize machine-readable notification formats that can allow firewalls and intrusion detection systems to act on them automatically? What role should coordinating bodies such as CERT play in this process? With additional work along these lines, automatic, measurement-driven mass notifications may someday be an important tool in the defensive security arsenal.

9. CONCLUSION

In this work we analyzed numerous aspects of the recent OpenSSL Heartbleed vulnerability, including (1) who was initially vulnerable, (2) patching behavior, and (3) impact on the certificate authority ecosystem. We found that the vulnerability was widespread, and estimated that between 24–55% of HTTPS-enabled servers in the Alexa Top 1 Million were initially vulnerable, including 44 of the Alexa Top 100. Sites patched heavily in the first two weeks after disclosure, but patching subsequently plateaued, and 3% of the HTTPS Alexa Top 1 Million sites remained vulnerable after two months. We further observed that only 10% of vulnerable sites replaced their certificates compared to 73% that patched, and 14% of sites doing so used the same private key, providing no protection.

We investigated the attack landscape, finding no evidence of large-scale attacks prior to the public disclosure, but vulnerability scans began within 22 hours. We observed post-disclosure attackers employing several distinct types of attacks from 692 sources, many coming from Amazon EC2 and Chinese ASes. We also conducted a mass notification of vulnerable hosts, finding a significant positive impact on the patching of hosts to which we sent notifications, indicating that this type of notification helps reduce global vulnerability. Finally, we drew upon our analyses to frame what went well and what went poorly in our community's response, providing perspectives on how we might respond more effectively to such events in the future.

Acknowledgments

The authors thank Ivan Ristic for providing historical data on TLS support, as well as Elie Bursztein, Paul Pearce, Hovav Shacham, Aashish Sharma, and Matthias Vallentin. We similarly thank the exceptional sysadmins at the University of Michigan for their help and support throughout this project.

This work was supported in part by the Department of Homeland Security Science and Technology Directorate under contracts D08PC75388, FA8750-12-2-0235, and FA8750-12-2-0314; the National Science Foundation under contracts CNS-0751116, CNS-08311174, CNS-091639, CNS-1111699, CNS-1255153, and CNS-1330142; DARPA award HR0011-12-2-005; and the Department of the Navy under contract N000.14-09-1-1042.

10. REFERENCES

[1] Alexa Top 1,000,000 Sites. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[2] Bitcoin Core Version History. https://bitcoin.org/en/version-history.

[3] Installing OpenDKIM. http://www.opendkim.org/INSTALL.

[4] Telnet Server with SSL Encryption Support. https://packages.debian.org/stable/net/telnetd-ssl.

[5] Install Ejabberd, Oct. 2004. http://www.ejabberd.im/tuto-install-ejabberd.

[6] Cassandra Wiki - Internode Encryption, Nov. 2013. http://wiki.apache.org/cassandra/InternodeEncryption.

[7] Android Platform Versions, Apr. 2014. https://developer.android.com/about/dashboards/index.html#Platform.

[8] Apple Says iOS, OSX and "Key Web Services" Not Affected by Heartbleed Security Flaw, Apr. 2014. http://recode.net/2014/04/10/apple-says-ios-osx-and-key-web-services-not-affected-by-heartbleed-security-flaw/.

[9] Heartbleed F.A.Q., 2014. https://www.startssl.com/?app=43.

[10] The Heartbleed Hit List: The Passwords You Need to Change Right Now, Apr. 2014. http://mashable.com/2014/04/09/heartbleed-bug-websites-affected/.

[11] HP Support Document c04249852, May 2014. http://goo.gl/AcUG8I.

[12] Is Openfire Affected by Heartbleed?, Apr. 2014. https://community.igniterealtime.org/thread/52272.

[13] June 2014 Web Server Survey, 2014. http://news.netcraft.com/archives/2014/06/06/june-2014-web-server-survey.html.

[14] NGINX and the Heartbleed Vulnerability, Apr. 2014. http://nginx.com/blog/nginx-and-the-heartbleed-vulnerability/.

[15] Official BTCJam Update, Apr. 2014. http://blog.btcjam.com/post/82158642922/official-btcjam-update.

[16] SSL Pulse, Apr. 2014. https://www.trustworthyinternet.org/ssl-pulse/.

[17] Tomcat Heartbleed, Apr. 2014. https://wiki.apache.org/tomcat/Security/Heartbleed.

[18] Wikimedia's Response to the "Heartbleed" Security Vulnerability, Apr. 2014. https://blog.wikimedia.org/2014/04/10/wikimedias-response-to-the-heartbleed-security-vulnerability/.

[19] Adobe. Heartbleed Update, Apr. 2014. http://blogs.adobe.com/psirt/?p=1085.

[20] M. Al-Bassam. Top Alexa 10,000 Heartbleed Scan—April 14, 2014. https://github.com/musalbas/heartbleed-masstest/blob/94cd9b6426311f0d20539e696496ed3d7bdd2a94/top1000.txt.

[21] Alienth. We Recommend that You Change Your Reddit Password, Apr. 2014. http://www.reddit.com/r/announcements/comments/231hl7/we_recommend_that_you_change_your_reddit_password/.

[22] B. Amann, M. Vallentin, S. Hall, and R. Sommer. Extracting Certificates from Live Traffic: A Near Real-Time SSL Notary Service. Technical Report TR-12-014, ICSI, Nov. 2012.

[23] AWeber Communications. Heartbleed: We're Not Affected. Here's What You Can Do To Protect Yourself, Apr. 2014. http://blog.aweber.com/articles-tips/heartbleed-how-to-protect-yourself.htm.

[24] Bitcoin. OpenSSL Heartbleed Vulnerability, Apr. 2014. https://bitcoin.org/en/alert/2014-04-11-heartbleed.

[25] Bro Network Security Monitor Web Site. http://www.bro.org.

[26] N. Craver. Is Stack Exchange Safe from Heartbleed?, Apr. 2014. http://meta.stackexchange.com/questions/228758/is-stack-exchange-safe-from-heartbleed.

[27] R. Dingledine. Tor OpenSSL Bug CVE-2014-0160, Apr. 2014. https://blog.torproject.org/blog/openssl-bug-cve-2014-0160.

[28] Dropbox Support. https://twitter.com/dropbox_support/status/453673783480832000, Apr. 2014. Quick Update on Heartbleed: We've Patched All of Our User-Facing Services & Will Continue to Work to Make Sure Your Stuff is Always Safe.

[29] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman. Analysis of the HTTPS Certificate Ecosystem. In *Proc. ACM Internet Measurement Conference*, Oct. 2013.

[30] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-Wide Scanning and its Security Applications. In *Proc. USENIX Security Symposium*, Aug. 2013.

[31] A. Ellis. Akamai heartbleed Update (V3), Apr. 2014. https://blogs.akamai.com/2014/04/heartbleed-update-v3.html.

[32] A. S. Foundation. CouchDB and the Heartbleed SSL/TLS Vulnerability, Apr. 2014. https://blogs.apache.org/couchdb/entry/couchdb_and_the_heartbleed_ssl.

[33] GoDaddy. OpenSSL Heartbleed: We've Patched Our Servers, Apr. 2014. http://support.godaddy.com/godaddy/openssl-and-heartbleed-vulnerabilities/.

[34] L. Grangeia. Heartbleed, Cupid and Wireless, May 2014. http://www.sysvalue.com/en/heartbleed-cupid-wireless/.

[35] S. Grant. The Bleeding Hearts Club: Heartbleed Recovery for System Administrators, Apr. 2014. https://www.eff.org/deeplinks/2014/04/bleeding-hearts-club-heartbleed-recovery-system-administrators.

[36] B. Grubb. Heartbleed Disclosure Timeline: Who Knew What and When. Apr. 2014. http://www.smh.com.au/it-pro/security-it/heartbleed-disclosure-timeline-who-knew-what-and-when-20140415-zqurk.html.

[37] L. Haisley. OpenSSL Crash with STARTTLS in Courier, May 2014. http://sourceforge.net/p/courier/mailman/message/32298514/.

[38] IBM. OpenSSL Heartbleed (CVE-2014-0160), May 2014. https://www-304.ibm.com/connections/blogs/PSIRT/entry/openssl_heartbleed_cve_2014_0160.

[39] Infusionsoft. What You Need to Know About Heartbleed, Apr. 2014. http://blog.infusionsoft.com/company-news/need-know-heartbleed/.

[40] Internal Revenue Service. IRS Statement on "Heartbleed" and Filing Season, Apr. 2014. http://www.irs.gov/uac/Newsroom/IRS-Statement-on-Heartbleed-and-Filing-Season.

[41] W. Kamishlian and R. Norris. Installing OpenSSL for Jabberd 2. http://www.jabberdoc.org/app_openssl.html.

[42] Litespeed Technologies. LSWS 4.2.9 Patches Heartbleed Bug, Apr. 2014. http://www.litespeedtech.com/support/forum/threads/lsws-4-2-9-patches-heartbleed-bug.8504/.

[43] S. Marquess. Of Money, Responsibility, and Pride, Apr. 2014. http://veridicalsystems.com/blog/of-money-responsibility-and-pride/.

[44] M. Masnick. Shameful Security: StartCom Charges People To Revoke SSL Certs Vulnerable to Heartbleed, Apr. 2014. http://www.techdirt.com/articles/20140409/11442426859/shameful-security-startcom-charges-people-to-revoke-ssl-certs-vulnerable-to-heartbleed.shtml.

[45] N. Mehta and Codenomicon. The Heartbleed Bug. http://heartbleed.com/.

[46] Microsoft. Microsoft Services unaffected by OpenSSL Heartbleed vulnerability, Apr. 2014. http://blogs.technet.com/b/security/archive/2014/04/10/microsoft-devices-and-services-and-the-openssl-heartbleed-vulnerability.aspx.

[47] MongoDB. MongoDB Response on Heartbleed OpenSSL Vulnerability, Apr. 2014. http://www.mongodb.com/blog/post/mongodb-response-heartbleed-openssl-vulnerability.

[48] K. Murchison. Heartbleed Warning - Cyrus Admin Passowrd Leak!, Apr. 2014. http://lists.andrew.cmu.edu/pipermail/info-cyrus/2014-April/037351.html.

[49] E. Ng. Tunnel Fails after OpenSSL Patch, Apr. 2014. https://lists.openswan.org/pipermail/users/2014-April/022934.html.

[50] M. O'Connor. Google Services Updated to Address OpenSSL CVE-2014-0160 (the Heartbleed Bug), Apr. 2014. http://googleonlinesecurity.blogspot.com/2014/04/google-services-updated-to-address.html.

[51] P. Ondruska. Does OpenSSL CVE-2014-0160 Effect Jetty Users?, Apr. 2014. http://dev.eclipse.org/mhonarc/lists/jetty-users/msg04624.html.

[52] OpenSSL Project Team. OpenSSL Security Advisory, Apr. 2014. http://www.mail-archive.com/openssl-users@openssl.org/msg73408.html.

[53] OpenSSL Project Team. OpenSSL Version 1.0.1g Released, Apr. 2014. http://www.mail-archive.com/openssl-users@openssl.org/msg73407.html.

[54] OpenVPN. OpenSSL Vulnerability—Heartbleed. https://community.openvpn.net/openvpn/wiki/heartbleed.

[55] Oracle. OpenSSL Security Bug—Heartbleed / CVE-2014-0160, Apr. 2014. http://www.oracle.com/technetwork/topics/security/opensslheartbleedcve-2014-0160-2188454.html.

[56] L. Padron. Important Read – Critical Security Advisory And Patch for OpenSSL Heartbleed Vulnerability, Apr. 2014. http://blog.zimbra.com/blog/archives/2014/04/important-read-critical-security-advisory-patch-openssl-heartbleed-vulnerability.html.

[57] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23-24):2435–2463, 1999.

[58] PayPal. OpenSSL Heartbleed Bug—PayPal Account Holders are Secure, Apr. 2014. https://www.paypal-community.com/t5/PayPal-Forward/OpenSSL-Heartbleed-Bug-PayPal-Account-Holders-are-Secure/ba-p/797568.

[59] W. Pinckaers. http://lekkertech.net/akamai.txt.

[60] M. Prince. The Hidden Costs of Heartbleed, Apr. 2014. http://blog.cloudflare.com/the-hard-costs-of-heartbleed.

[61] Publishers Clearing House. Stay Smart About The "Heartbleed" Bug With PCH!, Apr. 2014. http://blog.pch.com/blog/2014/04/16/stay-smart-about-the-heartbleed-bug-with-pch/.

[62] Rackspace. Protect Your Systems From "Heartbleed" OpenSSL Vulnerability, Apr. 2014. http://www.rackspace.com/blog/protect-your-systems-from-heartbleed-openssl-vulnerability/.

[63] Red Hat. How to Recover from the Heartbleed OpenSSL vulnerability, Apr. 2014. https://access.redhat.com/articles/786463.

[64] T. Saunders. ProFTPD and the OpenSSL "Heartbleed" Bug, May 2014. http://comments.gmane.org/gmane.network.proftpd.user/9465.

[65] B. Say. Bleedingheart Bug in OpenSSL, Apr. 2014. http://www.stunnel.org/pipermail/stunnel-users/2014-April/004578.html.

[66] R. Seggelmann, M. Tuexen, and M. Williams. Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension. IETF Request for Comments (RFC) 6520, February 2012. https://tools.ietf.org/html/rfc6520.

[67] N. Sullivan. The Results of the CloudFlare Challenge. Apr. 2014. http://blog.cloudflare.com/the-results-of-the-cloudflare-challenge.

[68] Tumblr. Urgent Security Update, Apr. 2014. http://staff.tumblr.com/post/82113034874/urgent-security-update.

[69] United States Postal Service. Avoiding Heartbleed, May 2014. https://ribbs.usps.gov/importantupdates/HeartbleedArticle.pdf.

[70] M. Wimmer. Removed Support for OpenSSL, 2007. https://jabberd.org/hg/amessagingd/rev/bcb8eb80cbb9.

[71] WordPress. Heartbleed Security Update, Apr. 2014. http://en.blog.wordpress.com/2014/04/15/security-update/.

[72] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage. When Private Keys Are Public: Results from the 2008 Debian OpenSSL Vulnerability. In *Proc. ACM Internet Measurement Conference*, Nov. 2009.

[73] ZDNet. Heartbleed Bug Affects Yahoo, OKCupid Sites, Apr. 2014. http://www.zdnet.com/heartbleed-bug-affects-yahoo-imgur-okcupid-convo-7000028213/.

[74] ZEDOinc. https://twitter.com/ZEDOinc/status/456145140503957504, Apr. 2014. Customers and partners: none of the ZEDO sites or assets are affected by Heartbleed.