Analysis of SSL Certificate Reissues and Revocations in the Wake of Heartbleed

Liang Zhang
Northeastern University
liang@ccs.neu.edu

David Choffnes
Northeastern University
choffnes@ccs.neu.edu

Dave Levin
University of Maryland
dml@cs.umd.edu

Tudor Dumitraş
University of Maryland
tdumitra@umiacs.umd.edu

Alan Mislove
Northeastern University
amislove@ccs.neu.edu

Aaron Schulman
Stanford University
aschulm@stanford.edu

Christo Wilson
Northeastern University
cbw@ccs.neu.edu

ABSTRACT

Central to the secure operation of a public key infrastructure (PKI) is the ability to *revoke* certificates. While much of users' security rests on this process taking place quickly, in practice, revocation typically requires a human to decide to reissue a new certificate and revoke the old one. Thus, having a proper understanding of how often systems administrators reissue and revoke certificates is crucial to understanding the integrity of a PKI. Unfortunately, this is typically difficult to measure: while it is relatively easy to determine when a certificate is revoked, it is difficult to determine whether and when an administrator *should have* revoked.

In this paper, we use a recent widespread security vulnerability as a natural experiment. Publicly announced in April 2014, the Heartbleed OpenSSL bug, potentially (and undetectably) revealed servers' private keys. Administrators of servers that were susceptible to Heartbleed should have revoked their certificates and reissued new ones, ideally as soon as the vulnerability was publicly announced.

Using a set of all certificates advertised by the Alexa Top 1 Million domains over a period of six months, we explore the patterns of reissuing and revoking certificates in the wake of Heartbleed. We find that over 73% of vulnerable certificates had yet to be reissued and over 87% had yet to be revoked three weeks after Heartbleed was disclosed. Moreover, our results show a drastic decline in revocations on the weekends, even immediately following the Heartbleed announcement. These results are an important step in understanding the manual processes on which users rely for secure, authenticated communication.

IMC'14, November 5–7, 2014, Vancouver, BC, Canada.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3213-2/14/11 ...$15.00.
http://dx.doi.org/10.1145/2663716.2663758.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.3 [**Computer-Communication Networks**]: Network Operations; E.3 [**Data Encryption**]: Public Key Cryptosystems, Standards

Keywords

Heartbleed; SSL; TLS; HTTPS; X.509; Certificates; Reissue; Revocation; Extended validation

1. INTRODUCTION

Secure Sockets Layer (SSL) and Transport Layer Security (TLS)[1] are the de-facto standards for securing Internet transactions such as banking, e-mail and e-commerce. Along with a public key infrastructure (PKI), SSL provides trusted identities via certificate chains and private communication via encryption. Central to these guarantees is that private keys used in SSL are not compromised by third parties; if so, certificates based on those private keys must be reissued and revoked to ensure that malicious third parties cannot masquerade as a trusted entity.

Importantly, the PKI uses a default-valid model where potentially compromised certificates remain valid until their expiration date or until they are revoked. Revocation, however, is a process that requires manual intervention from certificate owners and cooperation from clients that use these certificates. As a result, the practical security of the PKI is dependent on the speed with which certificate owners and SSL clients update their revocation lists, operations that occur at human timescales (hours or days) instead of computer ones (seconds or minutes). An important open question is: when private keys are compromised, how long are SSL clients exposed to potential attacks?

In this paper, we address this question using a recent widespread security vulnerability as a natural experiment. In mid-April 2014, an OpenSSL security vulnerability, Heartbleed, made it possible for attackers to inspect servers' memory contents, thereby potentially (and undetectably) revealing servers' private keys. Administrators of

[1]TLS is the successor of SSL, but both use the same X.509 certificates. Throughout the paper, we refer to "SSL clients" and "SSL certificates," but our findings apply equally to servers using both protocols.

servers that were susceptible to Heartbleed should have operated under the assumption than an attacker had already obtained their private keys, and therefore should have revoked their certificates and reissued new ones [5], ideally as soon as the vulnerability was publicly announced.

The scope of this vulnerability—it is estimated that up to 17% of all HTTPS web servers were vulnerable [22]—makes it an ideal case study for evaluating large-scale properties of SSL security in the face of private key compromise. While previous studies have measured how quickly and thoroughly administrators patch software vulnerabilities [25, 27, 35], we are, to the best of our knowledge, the first to study administration of certificates in the wake of a vulnerability. In particular, this paper focuses on certificate revocation and reissues in response to the public announcement of Heartbleed, both in terms of how quickly certificates are reissued and whether or not the certificates are eventually revoked.

Toward this goal, we make the following key contributions. First, we conduct a large-scale measurement study of SSL certificates in the wild using both data collected from public archives and through custom measurements conducted after Heartbleed was publicized. We focus on the Alexa Top 1 Million (Top-1M) domains, for which we find a total of 628,692 valid SSL certificates from 166,124 unique domains.

Second, we conduct measurements to determine which servers remain vulnerable to Heartbleed and which ones were previously vulnerable but are now patched. We develop a new SSL implementation fingerprinting technique that is able to determine if a host is running a version of OpenSSL that was vulnerable in the past. We cross-validate with direct measurements of the vulnerability (we find our technique has a false positive rate of only 1.9%) and conduct scans to compose a list of previously vulnerable hosts. We find that the most popular web sites were more likely to have at least one host vulnerable to Heartbleed, likely because they often have more hosts.

Third, we develop novel heuristics to identify which certificates have been reissued in direct response to Heartbleed, as opposed to other reasons such as certificate expiration or periodic reissues. This allows us to understand how administrators do (or do not) react to potential private key compromise. We observe that while vulnerable sites with a higher Alexa rank were more likely to reissue their certificates, the vast majority (73.3%) of vulnerable certificates had not been reissued fully three weeks after the vulnerability was announced. These vulnerable certificates come from more than 55,000 unique domains.

Fourth, we analyze certificate revocation behavior over time and across certificate owners. We find a sharp (up to 40-fold) increase in revocations per day after the Heartbleed announcement, but for the majority (60%) of reissued certificates, the previous (vulnerable) certificate was not revoked. For those that are revoked due to Heartbleed, we find more revocations in certificate revocation lists (CRLs) to have explanations (reason codes) than revocations unrelated to Heartbleed, and they appear in the CRLs more quickly than revocations not due to Heartbleed. Further, we examine the update frequency of CRLs to determine if Certificate Authorities (CAs), the entities that issue certificates, serve as a "bottleneck" for revocations (as it is the CA who maintains the CRL). We find that CRLs appear to be updated frequently, with over 95% of them being updated within the previous 24-hour period.

The remainder of this paper is organized as follows. In the next section, we provide background about SSL/TLS, PKIs, and the Heartbleed vulnerability. In §3 we describe our dataset and methodology for extracting valid certificates and determining Heartbleed vulnerability at servers. §4 presents the results of our analysis, where we identify the behavior of certificate reissuing and revocation on a large dataset of Alexa's Top-1M web sites. We summarize related work in §5 and conclude in §6.

2. BACKGROUND

Secure Sockets Layer (SSL) and Transport Layer Security (TLS) offer application-layer confidentiality and integrity, and are the basis of the vast majority of secure online communication. Through the use of a public key infrastructure (PKI), these protocols also allow clients to authenticate the servers with whom they communicate. In this section, we provide a brief background of SSL/TLS and PKIs relevant to our study, and describe the recent Heartbleed vulnerability.

2.1 Certificates

A certificate is, at its core, a signed attestation binding a *subject* to a public key. Certificates are signed by a Certificate Authority (CA), who in turn has its own certificate, and so on, terminating at self-signed *root certificates*. There is a logical *chain* of certificates—leading from a root certificate through zero or more *intermediate* certificates, to a *leaf* certificate—wherein the certificate at level i is signed with the private key corresponding to the certificate at level $i-1$, with the exception of the self-signed certificate at the root. In practice, the topology of certificates can be somewhat complex, with CAs signing one another's certificates [17], but such details are not pertinent to the study performed in this paper.

When a client visits a site that supports, say, HTTPS, that site sends its certificate chain to the client, who verifies the signatures from leaf to root. If the client can successfully validate each signature, and if the client trusts the root certificate—for instance by checking it against a set of certificates pre-installed in the browser or operating system—then the client infers that the subject in the leaf certificate truly is the owner of the public key.

The predominant format of certificates is X.509 [6], which includes considerably more information than just subject and public key, including a unique (for that CA) serial number, an expiration date, the key's cipher suite, acceptable uses of the key, and information on how to check whether the certificate has been *revoked*.

2.2 Certificate Revocation

In addition to issuing certificates, CAs are also responsible for making available a list of certificates it has issued that have been *revoked*, after which clients should no longer consider those certificates valid. Note that, if a CA's (intermediate or root) certificate is revoked, all leaf certificates signed by that CA will fail to validate.

There are many reasons a site can decide to revoke a certificate. One critically important example is that of a compromised certificate. A certificate is *compromised* if someone other than its original owner learns the corresponding private key, allowing that person to generate signatures and thus impersonate the owner. In the case of a CA certificate, release of the private key may allow an attacker to generate new certificates for arbitrary subjects. In such an event, it is important that the owner revokes the compromised certificate as quickly as possible to mitigate the set of users affected by the compromise.

Certificate Revocation Lists (CRLs) are by far the most common means of disseminating revocations. CRLs consist of a list of (serial number, time stamp of revocation, reason for revocation) triples, all of which are collectively signed by the CA. CAs include in the certificates that they issue a URL pointing to the CRL that would contain that certificate's serial number, if it were to become revoked. Clients periodically download and cache CRLs, and use them when validating a certificate chain. Ostensibly to reduce the communication overhead for CAs and for users, clients typically download CRLs infrequently (on the order of hours or days), potentially leaving many users with outdated information on the validity of their certificates. This has spurred several studies into more efficient means of revocation [12, 21, 23, 29, 36], and general doubt as to the overall efficacy of revocations [28]. Yet, CRLs remain the de facto means of disseminating revocation information, and thus they factor heavily in our study.

2.3 Certificate Reissues

When a site ceases to use a certificate—for instance because they found that the certificate has been compromised, or because the certificate expired—they must use a new certificate instead. This process is referred to as *reissuing* the certificate. To do so, the system administrator must contact the CA who signed their certificate and request a new signature; this is typically done by sending the CA a Certificate Signing Request (CSR). In the case where the private key may have been compromised, the administrator should also choose a new public/private key pair to be signed (as reissuing the certificate with the same key does nothing to mitigate the leaked private key).

While it seems natural to assume that certificates are reissued at precisely the moment the old certificate is revoked, in fact today's PKI protocols make no such requirement. As our study will demonstrate, reissues can happen before, during, or after a revocation—or even without revoking the old certificate at all. To the best of our knowledge, we are the first to correlate revocations with reissues.

2.4 Heartbleed

Heartbleed is a buffer over-read vulnerability discovered in OpenSSL [24] that was present in versions 1.0.1 (released March 14, 2012) through 1.0.1f. The vulnerability stems from a bug in OpenSSL's implementation of the TLS Heartbeat Extension [30]. The intended functionality of TLS Heartbeat is to allow a client to test a secure communication channel by sending a "heartbeat" message consisting of a string and the 16-bit `payload_length` of this string. Unfortunately, vulnerable OpenSSL versions fail to check that the `payload_length` supplied by the client matches the length of the provided string. This allows a malicious client to craft a heartbeat message containing a 1-byte string and $2^{16} - 1$ as the `payload_length`. In this case, OpenSSL will allocate a 64KB block of heap memory, `memcpy()` 64KB of data into it, starting with the 1-byte string, and finally send the contents of the entire buffer to the client. In effect, this allows the malicious client to read up to $2^{16} - 2$ bytes of the server's heap memory. Note that while the malicious client can choose the amount of memory to read, it has no control over the location of the memory that is copied, and therefore cannot choose *which* memory to read.

By repeatedly exploiting Heartbleed, an attacker can extract sensitive data from the server (e.g., SSL private keys [32], user data [13], etc.). The severity of Heartbleed is exacerbated by the fact that OpenSSL does not log heartbeat messages, giving attackers free reign to undetectably exploit Heartbleed. Given the severity and undetectable nature of malicious users exploiting Heartbleed, site operators were urged to immediately update their OpenSSL software and revoke and reissue their certificates [5].

Timeline. Heartbleed was first discovered by Neel Mehta from Google on March 21, 2014. Google immediately wrote a patch and applied it to their own OpenSSL deployments. On April 2, researchers at Finnish security company Codenomicon independently discovered the bug and dubbed it Heartbleed. On April 4, Akamai patched their servers. On April 7, the bug became public and the OpenSSL project released a patched version (1.0.1g) of the OpenSSL library [15].

Why study Heartbleed? The significance of this timeline, and of Heartbleed in general, is that it represents a point in time after which *all* vulnerable servers *should have* taken three critical steps to ensure the security of their service and their users: they should have patched their code, revoked their old certificate, and reissued a new one. As a result, Heartbleed acts as a sort of natural experiment, allowing us to measure how completely and quickly administrators took steps to secure their keys. While such events are (sadly) not terribly uncommon for general security vulnerabilities [25, 27, 35], it remains rare that such a large fraction of the certificate ecosystem must reissue and revoke their SSL certificates.

3. DATA AND METHODOLOGY

We now describe the data sets that we collected and our methodology for determining a host's SSL certificate, when it was in use, if and when the certificate was revoked, and if the host was (or is still) vulnerable to the Heartbleed bug.

3.1 Certificate Data Source

We obtain our collection of SSL certificates from (roughly) weekly scans of the entire IPv4 address space made available by Rapid7 [26]. In this paper, we use scans collected between October 30, 2013 and April 28, 2014. There are a total of 28 scans during this period, giving an average of 6.7 days (with a minimum of 3 days and maximum of 9 days) between successive scans.

The scan data includes *all certificates* advertised by each host (including intermediate and root certificates) in the scans up through February 5, 2014, and includes *only the first advertised certificate* by each host in the later scans. For example, suppose that a host is advertising a chain of three certificates: a certificate for `example.com`, a certificate for GeoTrust, and self-signed root certificate, where each certificate signs the previous. The earlier scans would include all three certificates, whereas the later scans would include only the certificate for `example.com`. The lack of full certificate chains in the later scans presents challenges for validation, which we address in §3.2.

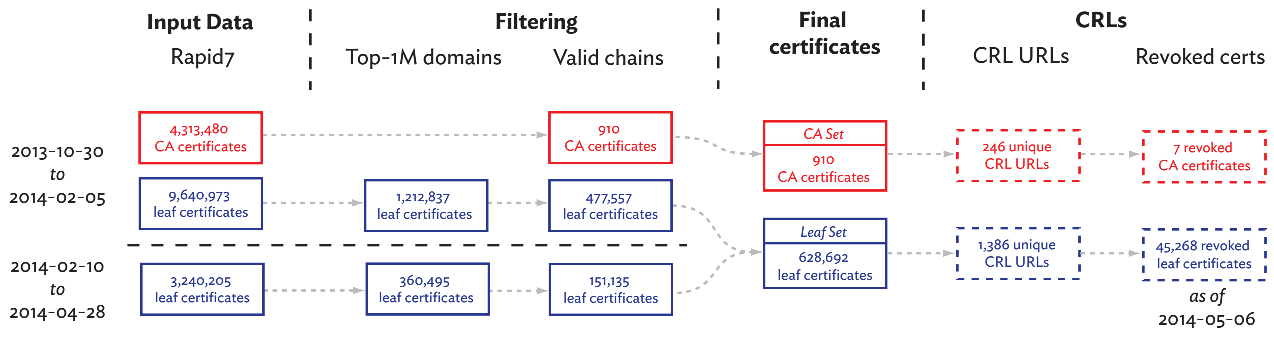
Input Data
Rapid7
Filtering
Top-1M domains
Valid chains
Final certificates
CRLs
CRL URLs
Revoked certs
2013-10-30 to 2014-02-05
4,313,480 CA certificates
9,640,973 leaf certificates
1,212,837 leaf certificates
910 CA certificates
477,557 leaf certificates
CA Set
910 CA certificates
246 unique CRL URLs
7 revoked CA certificates
2014-02-10 to 2014-04-28
3,240,205 leaf certificates
360,495 leaf certificates
151,135 leaf certificates
Leaf Set
628,692 leaf certificates
1,386 unique CRL URLs
45,268 revoked leaf certificates
as of 2014-05-06

Figure 1: Workflow from raw scans of the IPv4 address space to valid certificates (and corresponding CRLs) from the Alexa Top-1M domains. The Rapid7 data after February 5, 2014 did not include the intermediate (CA) certificates, necessitating additional steps and data to perform validation.

The scans found an average of 26.9 million hosts responding to SSL handshakes on port 443 (an average of 9.12% of the entire IPv4 address space). Across all of the scans, we observed a total of 19,438,865 unique certificates (including all leaf and CA certificates). In the sections below, we describe how we filtered and validated this data set; an overview of the process is provided in Figure 1.

3.2 Filtering Data

To focus on web destinations that are commonly accessed by users, we use the Alexa Top-1M domains [2] as observed on April 28, 2014. We first extract all leaf (non-CA) certificates that advertise a *Common Name* (CN) that is in one of the domains in the Alexa list (e.g., we would include certificates for `facebook.com`, `www.facebook.com`, as well as `*.dev.facebook.com`). This set represents 1,573,332 certificates (8.1% of all certificates). In order to remove invalid and self-signed certificates from this list, we then extract all advertised chains for these certificates (which are only present in the scans through February 5, 2014).

Reconstructing chains. The lack of full certificate chains for the post-February 5, 2014 scans (see §3.1) presents a challenge at this point, as we need the full certificate chains in order to properly validate the leaf certificates. To verify new certificates observed in these later scans, we construct a list of all 4,509 intermediate (CA), non-self-signed certificates observed in previous scans.[2] From these certificates, we use two types of X.509 fields to help with chain reconstruction [6]:

- The *Subject Key Identifier* and *Authority Key Identifier* are two fields included in most certificates, and uniquely identify the public key the certificate represents (*Subject Key Identifier*) and the public key that signed this certificate (*Authority Key Identifier*). The value is typically implemented as a hash of the public key.
- The *Subject Name* and *Issuer Name* are text fields that represent the name of the entity this certificate represents (*Subject Name*) and the name of entity that signed this certificate (*Issuer Name*).

[2]We also conduct our own crawl (see §3.4) of hosts advertising certificates in the Alexa list, and included all 4,445 additional non-self-signed CA certificates that we discovered in this list as well. However, we found that none of the additional CA certificates were necessary for validation.

We construct a database of all four of these fields across all 8,954 CA certificates.

Using this database, we attempt to reconstruct a leaf certificate's chain based first on the certificate's *Authority Key ID* and, failing that, the certificate's *Issuer Name*. In other words, given a leaf certificate, we look for a CA certificate whose *Subject Key Identifier* is the same as our leaf's *Authority Key Identifier*. Should we not find one (or should the *Subject Key Identifier* not be present), we instead look for a CA certificate whose *Subject Name* is the same as our leaf's *Issuer Name*. We then recursively apply this technique until we cannot find a parent key, we hit a trusted root certificate, or we hit a self-signed CA certificate. Should we find multiple CA keys that match at any stage, we include them all as potential chains.

Verifying chains. We then unify our set of potential chains, consisting of both host-advertised chains (for the data collected through February 5, 2014) and reconstructed-chains (for the data collected post-February 5, 2014). Unfortunately, despite the leaf certificate having a *Common Name* in the Alexa list, many of our chains may not be valid (e.g., expired certificates, forged self-signed certificates, certificates signed by an invalid root, etc.). One common source of invalid certificates is home routers/DSL modems provided by ISPs (e.g., FRITZ!Boxes) or cloud-accessible storage devices (e.g., Western Digital's My Cloud), both of which advertise self-signed SSL certificates in the `fritz.net` and `wd2go.com` domains.

We removed these invalid chains by running `openssl verify` on each certificate (and its corresponding chain), and only kept the certificates that OpenSSL could verify. Because the scans occurred at different points of time, we used the `faketime` library [14] to have OpenSSL validate the certificate as of the time of the scan. We also configure OpenSSL to trust the set of root CA certificates included by default in the OS X 10.9.2 root store [20]; this includes 222 unique root certificates.

After validation, we are left with 628,692 leaf certificates (40.0% of all certificates advertising Alexa domains and 3.2% of all certificates) from Alexa Top-1M domains that were advertised by some IP address and could be validated; we refer to this set of certificates as the *Leaf Set*. Each of these

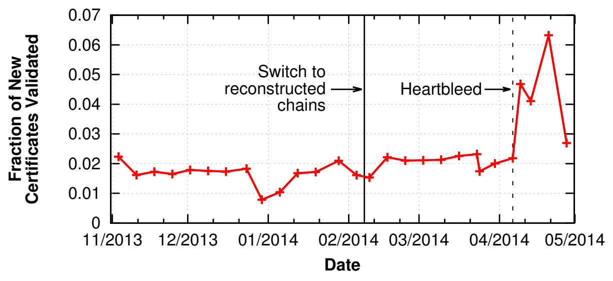

Figure 2: Fraction of new certificates that we could verify for provided (February 5, 2014 and before) and reconstructed (post February 5, 2014) chains.

certificates has a valid chain; we refer to the collection of all CA certificates on these chains (not including the leaf certificates) as the *CA Set*; the CA Set contains 910 unique certificates. The Leaf Set certificates cover 166,124 (16.6%) of the Alexa Top-1M domains. This is the set of certificates (and certificate chains) that we use in the remainder of the paper.

Validation of reconstruction. Finally, we briefly validate our certificate chain reconstruction mechanism on the post-February 5, 2014 certificates. In Figure 2, we present the fraction of new certificates discovered over time for which we were able to find a valid chain, both for the pre- and post-February 5, 2014 data. We make two interesting observations: *First*, the fraction of certificates that we could validate is relatively stable at 2% both before and after the switch to using reconstructed chains, suggesting that our mechanism for chain reconstruction does not miss many chains. *Second*, we see a large uptick in the fraction of newly-appearing certificates that we could validate after Heartbleed; as we discuss in the following section, this is due to many certificates being reissued in the wake of Heartbleed.

3.3 Collecting CRLs

To determine if and when certificates were revoked, we extracted the CRL URLs out of all Leaf Set certificates. We ignored invalid URLs, including `ldap://` protocols and non-routable addresses. We found 626,659 (99.7%) of these certificates to include at least one well-formed, reachable CRL URL; for certificates that included multiple CRL URLs, we included them all. We found a total of 1,386 unique CRL URLs (most certificates use a unified CRL provided by the signing CA, so the small number of CRLs is not surprising). We downloaded all of these CRLs on May 6, 2014, and found 45,268 (7.2%) of the Leaf Set certificates to be revoked.

We also collected the CRL URLs for all certificates in the CA set. We found that 884 (97.1%) of the certificates in the CA Set included a reachable CRL; the union of these URLs comprised 246 unique reachable URLs. We downloaded these CRLs on May 6, 2014, as well. We found a total of seven CA certificates that were revoked, which nullified the validity of 60 certificates in the Leaf Set ($< 0.01\%$).

3.4 Inferring Heartbleed Vulnerability

Finally, we wish to determine if a site was ever vulnerable to the Heartbleed OpenSSL vulnerability (and if it continued to be vulnerable at the end of the study). Doing so allows us to reason about whether the site operators should have reissued their SSL certificate(s) and revoked their old one(s). Determining if a host is currently vulnerable to Heartbleed is relatively easy, as one can simply send improperly-formatted SSL heartbeat messages to test for vulnerability.

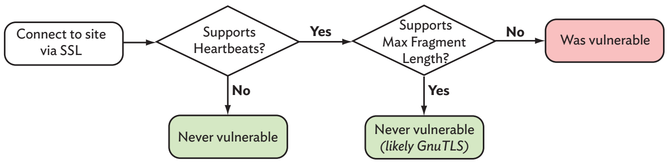

Figure 3: Flowchart of inference of previous Heartbleed vulnerability of hosts based on our SSL scan.

However, determining if a site *was* vulnerable at some point in the past—but has since updated their OpenSSL code—is more challenging. We observe that only three of the common TLS implementations have ever supported SSL Heartbeats [30]: OpenSSL [24], GnuTLS [33], and Botan [4]. Thus, if a host supports the SSL Heartbeat extension, we know that it is running one of these three implementations. Botan is a library that is targeted for client-side TLS, and we know of no popular web server that is able to use the Botan TLS library. GnuTLS has support for the SSL Heartbeat extension, but it is not enabled by default.[3] To determine if the host is using GnuTLS, we observed that GnuTLS supports the Max Fragment Length SSL extension [1], which *is* enabled by default, while OpenSSL has never supported this extension. Thus, if we observe a host that supports the SSL Heartbeat extension but *not* the Max Fragment Length extension, we declare that host to have been running a version of OpenSSL that was vulnerable (see Figure 3 for a graphical representation).

To collect the list of sites that were ever vulnerable to Heartbleed, we first extracted the set of IP addresses in the April 28, 2014 Rapid7 scan that were advertising a certificate with a *Common Name* in the Alexa Top-1M list. We found 5,951,763 unique IP addresses in this set. We then connected to these IP addresses, performed the TLS negotiation, determined the SSL extensions that the host supported, and determined whether the host was still vulnerable to the Heartbleed vulnerability. We also downloaded the set of CA certificates that the host advertised, which we used to aid certificate validation (see §3.2).

Limitations. Our methodology for inferring a host's vulnerability to Heartbleed has the following limitations. Because we did our scan three weeks after Heartbleed was announced, we may have both false positives and false negatives in detecting whether a host was ever vulnerable to Heartbleed. For false positives, hosts that were upgraded *directly* from OpenSSL 0.9.8 to OpenSSL 1.0.1g (i.e., bypassing the Heartbleed bug) would be incorrectly flagged as being vulnerable in the past. We suspect this fraction is small, as this would have had to have happened between April 7th (the release of OpenSSL 1.0.1g) and April 28th (our scan), but we are unable to estimate the fraction of hosts this covers.

For false negatives, administrators who responded to Heartbleed by either recompiling OpenSSL with

[3]In fact, in our scan, we did not discover *any* hosts that were running GnuTLS with SSL Heartbeats enabled.

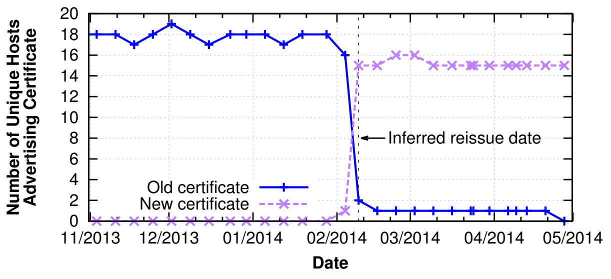

Figure 4: Example of lifetime, for certificates for `m.scotrail.co.uk`. All hosts except one switch to a new certificate after February 10, 2014.

`-DOPENSSL_NO_HEARTBEATS` or who downgraded their OpenSSL implementation to version ≤0.9.8 would have their hosts incorrectly flagged as never having been vulnerable. We are similarly unable to determine the fraction of hosts in our data set that this applies to; we suspect it is small as well, as many operating systems vendors (e.g., Ubuntu) pushed out a Heartbleed security update that is usually automatically applied.

Verification of vulnerability detection. We performed a brief experiment to estimate the false negative rate of our Heartbleed vulnerability detection mechanism. We use a vulnerability scan of the Alexa Top-1M domains conducted by the authors of ZMap [37] on April 9, 2014, which contains a list of hosts they confirmed to be vulnerable to Heartbleed. In our scan on April 28, 2014 (19 days after the ZMap scan), we found that 8,651 of these hosts were still advertising a certificate with the same *Common Name*. Of these, 1,737 (20.1%) were *still* vulnerable; the remainder were likely patched in the meantime. Using our fingerprinting methodology above, we would have inferred that 8,483 (98.1%) of the hosts were running a version of OpenSSL that was vulnerable at some point (despite the fact that the majority of these were actually no longer vulnerable). This high rate of recall, coupled with the unlikelihood of false negatives, leads us to conclude that our methodology for inferring previous vulnerability is highly accurate.

4. ANALYSIS

We now turn to examine the collected SSL certificate data. We first present a few definitions we use in the analysis before proceeding.

4.1 Definitions

We are concerned with the evolution of SSL certificates (i.e., when are new certificates created, old ones retired, etc.). To aid in understanding this evolution, we define the following notions:

Certificate birth: We define the birth of an SSL certificate to be the date of the first scan where we observed any host advertising that certificate. For hosts that we observed advertising a certificate on the very first scan (October 30, 2013), we define these certificates to have no birth date, since we do not know when they were first advertised.[4]

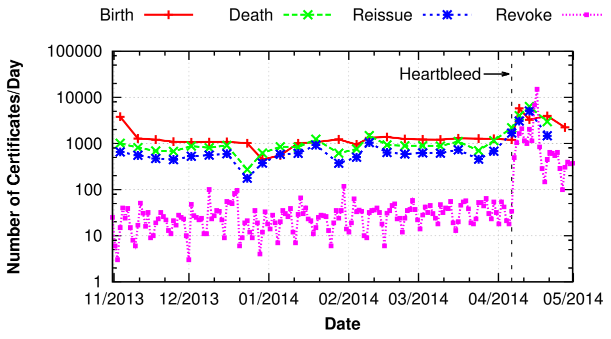

Figure 5: Number of certificate birth, deaths, reissues, and revocations over time. Note the log scale on the y-axis.

Certificate death: Defining the death of a certificate is more complicated, as we observe a number of instances where many hosts advertise a given certificate, and then all but one or a few of the hosts switch over to a new certificate (presumably, the site intended to retire the old certificate, but missed some of the hosts). To handle these cases, we calculate the *maximum* number of hosts that were ever advertising each certificate. We then define the death of an SSL certificate to be the last date that the number of hosts advertising the certificate was above 10% of that certificate's maximum. The 10% threshold prevents us from incorrectly classifying certificates that are still widely available as dead, even if the certificate has been reissued. Note that certificates may not have a death date if the certificate is still advertised by many IP addresses on our last scan.

An example of certificate lifetime is shown in Figure 4, for the certificates for `m.scotrail.co.uk`. All hosts except one switch to a new certificate after February 10, 2014; this lone host finally switches on April 28, 2014. In this case, we would consider the death date of the old certificate to be February 10, 2014 (as indicated in the figure), and we would consider the new certificate to have no death date.

Based on these definitions, we can now define the notion of a certificate reissue and revocation:

Certificate reissue: We consider a certificate to be reissued if the following three conditions hold: (a) we observe the certificate die, and (b) we observe a new certificate for the same *Common Name* born during a scan within 10 days[5] of the certificate's death, and (c) we observe at least one IP address switch from the old certificate to the new between the two scans. We define the date of the certificate reissue to be the date of the certificate's death. For the sake of clarity, we refer to the old certificate that was replaced as the *retired certificate*.

Certificate revocation: We consider a certificate to be revoked if the certificate's serial number appears in any of the certificate's CRLs. The date of revocation is provided in the CRL entry.

[4]Of course, some certificates may have been missed on the first scan if the host was down; these certificates would likely show up in the second scan (and would have a birth date of the next scan). This is the cause of the small spike in births on November 2, 2013 in Figure 5.

[5]We choose 10 days as a threshold as this is the maximum difference between two successive scans.

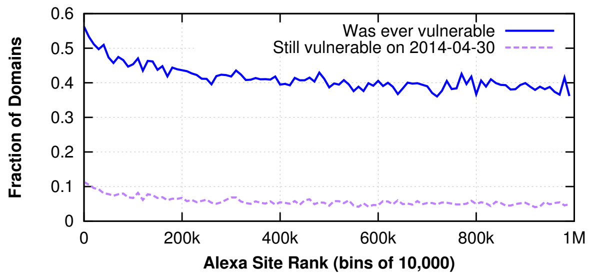

Figure 6: Fraction of domains that have at least one host that was ever vulnerable to Heartbleed as a function of Alexa rank, as well as domains that continued to be vulnerable at the end of the study.

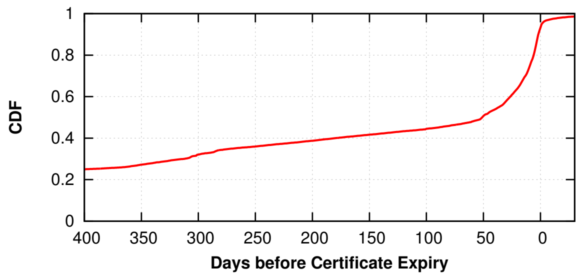

Figure 7: Cumulative distribution of the number of days before expiration that certificates are reissued.

In Figure 5, we present the number of certificate births, deaths, reissues, and revocations per day over time. The number of births is almost always larger than the number of deaths, meaning that the total number of certificates in-the-wild is increasing over time. Furthermore, we observe a large spike in all four events in the wake of Heartbleed, with an especially large increase in the number of revocations. For example, we see an average of 29 certificate revocations per day before Heartbleed; after Heartbleed, this jumps to an average of 1,414 revocations per day.

4.2 Heartbleed Prevalence

We present a brief analysis on the number of certificates hosted by machines that were ever vulnerable to Heartbleed. Of the 428,552 leaf certificates that were still alive on the last scan, we observe 122,832 (28.6%) of them advertised by a host that was likely vulnerable to Heartbleed at some point in time.[6] These certificates are for 117,112 unique *Common Name*s and come from 70,875 unique Alexa Top-1M domains. Of these certificates, 11,915 certificates (from 10,366 unique domains) were on hosts that were *still* vulnerable at the time of our crawl (April 30, 2014, over three weeks after the announcement of Heartbleed). This result demonstrates that even in the wake of a well-publicized, severe security vulnerability, around 10% of vulnerable sites have not yet addressed the underlying issue three weeks after the fact.

In Figure 6, we present the fraction of domains that have at least one SSL host that was ever vulnerable to Heartbleed (or still was as of April 30, 2014). We can observe a slight increase in likelihood of ever being vulnerable for the most popular sites, but the distribution quickly stabilizes. Again, the increased likelihood of being vulnerable is likely because these sites have larger numbers of hosts. This trend is mirrored in the hosts that are still vulnerable on April 30, 2014.

4.3 Certificate Reissues

We now examine the reissuing of SSL certificates in the wake of Heartbleed. Not all SSL certificate reissues that we observe following Heartbleed's announcement are due to the Heartbleed vulnerability. In particular, reissues can happen for at least two other reasons: *First*, the old certificate could be expiring soon, and the organization reissues the certificate as it would normally. In Figure 7, we present the cumulative distribution of the number of days before expiry that we observe certificates being reissued. We see that over 50% of certificates are reissued within 60 days of their expiry date (with a long tail).

Second, a site may periodically reissue certificates as a matter of policy (even if the old certificate was not near expiration). For example, Figure 8 presents a graph showing the prevalence of the `www.google.com` certificates over time, with each line representing the number of hosts advertising a different certificate. Google typically reissues this certificate every two weeks, despite the fact that the certificates are typically valid for more than three months.

In this study, we would like to be able to distinguish a *Heartbleed-induced* certificate reissue from a reissue that would otherwise have happened anyway. We define the reissue of a certificate to be Heartbleed-induced if all three of the following conditions hold:

1. The date of reissue was on or after April 7, 2014 (the day Heartbleed was announced). We note that a small number of organizations were informed about Heartbleed before the public announcement; as this list is not fully known, we do not consider them separately.

2. The certificate that is reissued was going to expire more than 60 days after the reissue. This eliminates certificates that were very likely to be reissued in the near future anyway.

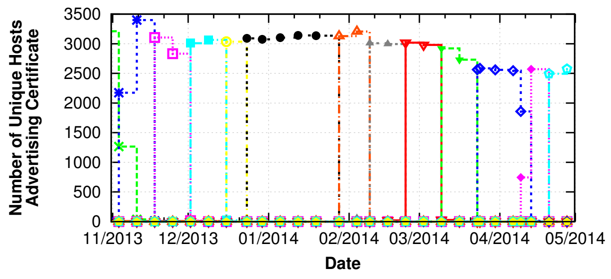

Figure 8: Example of certificate birth and death for certificates for `www.google.com`. Google reissues this certificate about once every two weeks (each impulse represents a different certificate).

[6]This fraction is somewhat higher than the 17% of sites that Netcraft found to be vulnerable [22], but we note that we are measuring *certificates* from the Alexa Top-1M while Netcraft is measuring all SSL-enabled *sites* on the Internet.

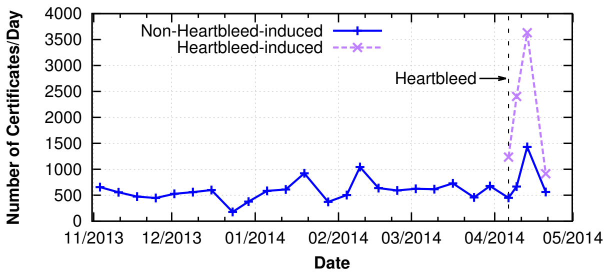

Figure 9: Number of Heartbleed-induced and non-Heartbleed-induced certificate reissues over time.

3. We do not observe more than two other reissues for certificates with that *Common Name* in the time before Heartbleed. This implies that certificates with that name do not typically get reissued more than once every 3 months (as far as we can observe from our dataset), as our data set begins on October 30, 2013 (slightly over 5 months before the announcement of the Heartbleed vulnerability).

Thus, for the examples shown so far, we would not have considered the reissue of the retired certificate in Figure 4 to be Heartbleed-induced (as it happened before Heartbleed), and we would also have not considered any of Google's reissues in Figure 8 to be Heartbleed-induced (because we observed a total of 12 reissues of certificates with that *Common Name* prior to Heartbleed). It is important to note that Heartbleed-induced reissues can happen for certificates that we never observed on a vulnerable host, either because we falsely declared the certificate to not be vulnerable (see §3.4) or because the site reissued out of an abundance of caution, even though they were not actually vulnerable. Given these three conditions, we expect that our estimate of Heartbleed-induced reissues is a strict *lower bound*.

Heartbleed-induced reissues. Overall, we observe 36,781 certificate reissues that we declare to be Heartbleed-induced in the three weeks following the announcement; this is 8.9% of all certificates that were alive at the time Heartbleed was announced. In Figure 9, we present the number of Heartbleed-induced and non-Heartbleed-induced certificate reissues over time. We observe that the number of non-Heartbleed-induced reissues is relatively stable—even after Heartbleed—suggesting our designation of Heartbleed-induced reissues is likely accurate. The slight spike in non-Heartbleed-induced reissues after April 7 may reflect that our approach yields a conservative underestimate of the number of Heartbleed-induced reissues.

Next, we examine the fraction of sites that have at least one Heartbleed-induced certificate reissue, as a function of Alexa rank. Figure 10 presents these results; we can observe a strong correlation with Alexa rank. Higher-ranked sites are much more likely to have reissued at least one certificate due to Heartbleed (even though they are only slightly more likely to have been vulnerable, as observed in Figure 6). This result complements previous studies' findings that more popular websites often exhibit more sound administrative practices [8, 17].

Vulnerable certificates. Next, we examine the certificates that *should have* been reissued (regardless of whether they

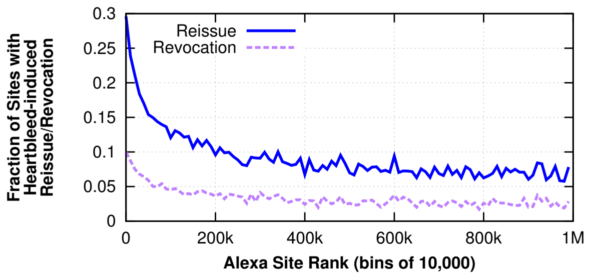

Figure 10: Fraction of domains that have at least one Heartbleed-induced reissue/revocation as a function of Alexa rank.

actually were); we refer to these certificates as *vulnerable certificates*. We declare a certificate to be vulnerable if the following three conditions hold:

1. Its date of birth was before April 7, 2014,
2. It has not expired as of April 30, and
3. It was advertised by at least one host that was (or is) vulnerable to Heartbleed.

In other words, these certificates are vulnerable because their associated private keys could have been stolen by attackers.

Overall, we find 107,712 vulnerable certificates. Of these, we observe that only 28,652 (26.7%) have been reissued as of April 30. The remaining 79,060 (73.3%) vulnerable certificates that have not been reissued come from 55,086 different Alexa Top-1M domains. Thus, the vast majority of SSL certificates that were potentially exposed by the Heartbleed bug remain in-use over three weeks after the vulnerability was announced.

Reissues with same key. System administrators who believe that their SSL private key may have been compromised should generate a new public/private key pair when reissuing their certificate. We now examine how frequently this is done, both in the case of normal certificate reissues and for Heartbleed-induced reissues.

We first observe that, in general, reissuing a certificate using the same public/private key pair is quite common. Figure 11 presents the fraction of all new certificates that use the same key as the one they are replacing; up to 53% of all reissued certificates do so. This high level of key reuse is

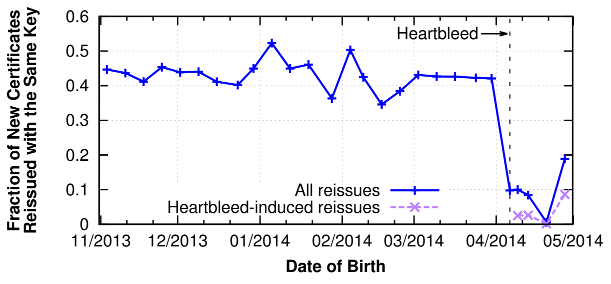

Figure 11: Fraction of new certificates that use the same public/private key pair as the key they are replacing.

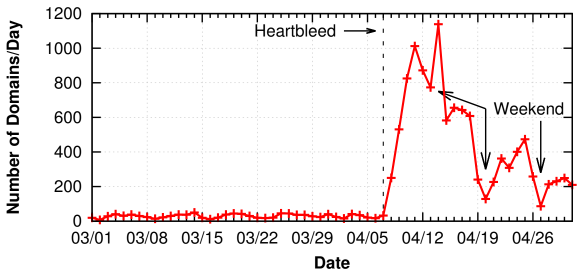

Figure 12: Number of domains that revoked at least one certificate over time for the month before and after Heartbleed.

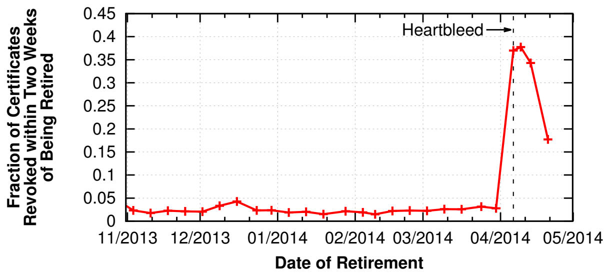

Figure 13: Fraction of reissued certificates that are revoked within two weeks of being retired. A significant increase in revocation probability is observed after Heartbleed.

at least partially due to system administrators re-using the same Certificate Signing Request (CSR) when requesting the new certificate from their CA.

In the wake of Heartbleed, we observe a significant drop in the frequency of reissuing certificates with the same key; this result indicates that sites are generating a new key pair more frequently. However, if we focus on the Heartbleed-induced reissues, we observe that a non-trivial fraction (4.1%) of these certificates are reissued with the same key (thereby defeating the purpose of reissuing the certificate). In fact, we observe a total of 912 such certificates coming from 747 distinct Alexa domains.

4.4 Certificate Revocation

We now turn to investigating certificate revocation before, during, and after the revelation of Heartbleed. Recall that it is critical that a vulnerable certificate be revoked: even if a site reissues a new certificate, if an attacker gained access to the vulnerable certificate's private key, then that attacker will be able to impersonate the owner until either the certificate expires or is revoked.[7] We study both revocation and expiration here, and correlate them with rates of reissue. Contrary to standard assumptions, we find that revocation and reissues do not happen simultaneously.

Overall revocation rates. Figure 5 shows the number of certificate revocations over time; as noted above, the average jumps from 29 certificates revoked per day to 1,414 post-Heartbleed. However, the spike on April 16, 2014 is somewhat misleading, as it was largely due to the mass-revocation of 19,384 CloudFlare certificates of the form `ssl XXXXX.cloudflare.com` [31].

To mitigate this issue, we plot in Figure 12 the number of unique *domains* that revoked at least one certificate over time. We make three interesting observations: *First*, the magnitude of the Heartbleed-induced spike is greatly reduced, but we still observe an up-to-40-fold increase in the number of domains issuing revocations per day. *Second*, we observe that the number of domains issuing revocations falls closer to its pre-Heartbleed level by April 28th, suggesting that *most of the domains that will revoke their certificate in direct response to Heartbleed already have.*

[7]We note that revocation alone is often insufficient to prevent impersonation, as an attacker may be able to prevent the client from accessing the CRL. In this case, many web browsers still accept the certificate as valid [18].

Third, we observe three "dips" in the post-Heartbleed revocation rate on April 13th, April 20th, and April 27th—all weekends, indicating that far fewer revocations occur on the weekend relative to the rest of the week. This periodicity can also be (less-easily) observed in the pre-Heartbleed time frame. It is reasonable to assume revocations dip on weekends because humans are involved in the revocation process, however it is not clear who is responsible for the delays: is it site administrators or CRL maintainers at CAs (or both) who are not working on weekends? Regardless of who is responsible, these weekend delays are problematic for online security, since vulnerabilities (and the attackers who exploit them) do not take weekends off.

Revocation of reissued certificates. We now examine the fraction of retired certificates (i.e., old certificates that have been superseded by a reissued cert) that are revoked within two weeks of being retired. Figure 13 plots this fraction over time. For example, the point on March 3, 2014 shows that 2.2% of the certificates retired on that day were revoked by March 17, 2014. Overall, we see that between 2% and 3% of certificates being retired are eventually revoked. This probability increases by an order of magnitude after Heartbleed, with almost 40% of retired certificates being revoked quickly afterwards. This result suggests that the reason many certificates were reissued just after April 7 was because of Heartbleed, since the retired certificates were also revoked. This contrasts with certificates that are reissued due to impending expiration, in which case the retired certificate does not need to be revoked.

Heartbleed-induced revocations. Similar to certificate reissues, not all certificate revocations after April 7, 2014 are necessarily due to Heartbleed (e.g., the site could have exposed their private key due to a different vulnerability). We therefore define a *Heartbleed-induced revocation* to be a certificate revocation where the certificate had a Heartbleed-induced reissue (see §4.3).

Overall, we observe 14,726 Heartbleed-induced revocations; this corresponds to 40% of all Heartbleed-induced reissued certificates. Thus, 60% of all certificates that were reissued due to Heartbleed were *not* revoked, implying that, if the certificate's private key was actually stolen, the attacker would be able to impersonate the victim without any clients being able to detect it.

Figure 10 presents the fraction of sites that have at least one Heartbleed-induced certificate revocation, as a function of Alexa rank. Revocations follow a similar trend to reis-

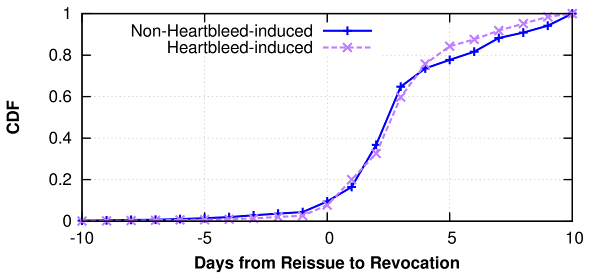

Figure 14: Cumulative distribution of the number of days between when a certificate is reissued and when it is revoked. Positive values indicate the certificate is reissued before it is revoked; negative values indicate the opposite.

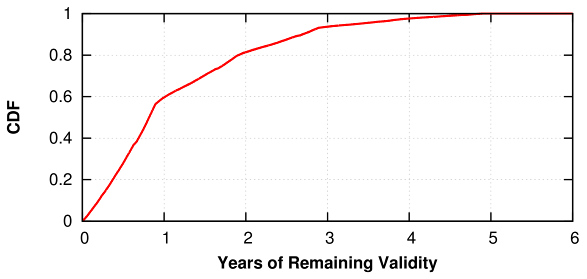

Figure 15: The distribution of time-until-expiry for vulnerable, reissued, but not revoked certificates. If these certificates are never revoked, this figure shows how long they will persist.

sues, i.e., sites with high rank are slightly more likely to revoke. Ideally, the two lines in Figure 10 should be coincident, i.e., all sites reissuing certificates due to Heartbleed should also have revoked the retired certificates (the only exception to this rule is if the retired certificate was about to expire anyway, but we account for this in our definitions of Heartbleed-induced reissues and revocations). This result highlights a serious gap in security best-practices across all of the sites in the Alexa Top-1M.

Finally, we examine the *revocation speed*, or the number of days between when a certificate is reissued and it is revoked. Figure 14 presents the cumulative distribution of the revocation speed for both Heartbleed-induced and non-Heartbleed-induced revocations. To make the distributions comparable, we only look at differences between -10 and 10 days (recall that Heartbleed-induced reissues and revocations can only occur after April 7, 2014, limiting that distribution). We observe that Heartbleed-induced revocations appear to happen slightly more quickly, thought not to the extent one might expect, given the urgent nature of the vulnerability. We also observe that revocation almost always happens *after* reissue, which is likely explained by the more manual process that revocation often entails. This result contradicts previous assumptions [8] that revocations and reissues occur simultaneously. Finally, it is worth noting that the granularity of our scans makes generalizing these results difficult, since we cannot tell exactly when a certificate was reissued; however, the two distributions are comparable to each other.

Expirations are not enough. To demonstrate how long the effects of this vulnerability could be felt if sites do not revoke their vulnerable certificates, we analyze certificates that, by the end of our data collection, were found to be vulnerable (and alive) when Heartbleed was announced, reissued thereafter, but never revoked. Figure 15 presents the distribution of how much longer such certificates will continue to live if their sites do not revoke them. Note that this CDF appears to be piecewise linear at intervals of 1 year: this is because expiration dates are typically set at intervals of a year—that the distribution is roughly uniform within these year intervals indicates that certificates are issued mostly uniformly throughout the year. This figure shows that, without revoking, the vulnerability introduced in 2014 could affect clients through 2020. We conclude from this that, given the meager rates of revocation, it would be helpful for CAs to shift to shorter expiry times in their certificates.

CRL reason codes. The CRL specification allows the maintainers of CRLs to include a *reason* for why a certificate was revoked along with the revocation in the form of a small set of *reason codes*. The reason code is optional, and the options range from "Unspecified" to "Key Compromised" to "Privilege Withdrawn" [6]. Note that the CRL reason codes are not necessarily verified by the certificate authorities, and they may be incorrect.

For all of the certificates that we observed to be revoked, we extracted the reason code (if one existed); we present the distribution of these reason codes for both Heartbleed-induced and non-Heartbleed-induced certificate reissues in Figure 16. Note the log-scale on the x-axis.

We make two key observations. *First*, we see a significant increase in the probability of a reason code being provided at all for Heartbleed-induced revocations: only 19.2% of non-Heartbleed-induced revocations provide any reason code (*including* the "Unspecified" reason code), while 27.1% on Heartbleed-induced revocations provide a reason code. *Second*, we observe a large increase in the "Key Compromise" reason code (from 0.40% to 1.18% of all CRL entries); given that Heartbleed certificates are likely being reissued

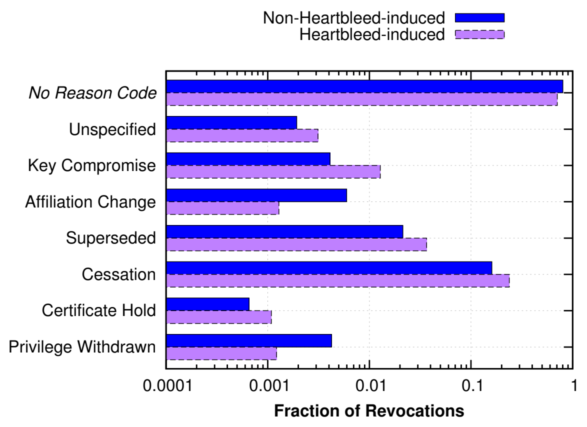

Figure 16: Distribution of CRL reason codes given for both Heartbleed-induced and non-Heartbleed-induced certificate reissues. Note the log scale on the x-axis. We observe an increase in reasons for revocations being given for Heartbleed-induced reissues, especially for the "Key Compromised" reason code.

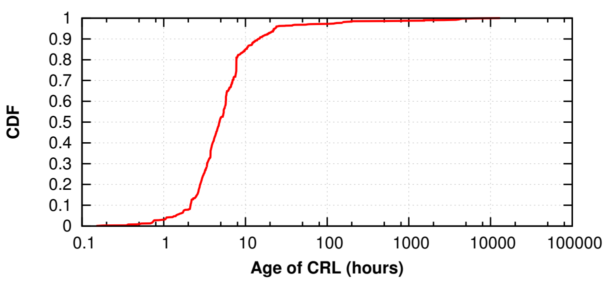

Figure 17: Cumulative distribution of the time between when we downloaded the CRLs (6:00pm EST) and the time of issue recorded in the CRL (and signed by the CA). Most CAs have a chance to revoke certificates at least once a day, as 95% of the CAs updated their CRLs within 24 hours of when we downloaded them.

due to concerns that the private key may have been compromised, this increase is not unexpected. However, it still appears that vast majority of CRL entries are mis-coded. Prior work has also noted that CRLs are usually mis-coded [8], although the snapshot we present in Figure 16 is even more stark, given that we know Heartbleed-induced revocations should have been revoked with a reason code of "Key Compromise".

CRL update intervals. The general lack of site administrators revoking certificates when they should (e.g., after Heartbleed) could be attributed to the CAs only updating their CRLs on very long timescales. For example, one reason for this would be if CAs kept their private keys on offline hosts that would have to be powered on every time to sign CRLs. Another reason would be so clients do not need to download new CRLs very often.

Figure 17 indicates that neither of these reasons are true. This figure shows the cumulative distribution of the difference between the time we downloaded a CRL and the time it was issued. We see that 95% of CAs signed a fresh CRL within 24 hours of 6:00pm EST (when we downloaded the CRLs). When CAs sign a fresh CRL, they have the opportunity to revoke more certificates. These results suggest that CAs could revoke certificates as often as every few hours. Thus, any delays in the revocation of certificates are due to humans in the loop: either certificate owners who are not reporting potentially compromised keys, or CA personnel who are not manually adding new entries to CRLs before they are signed and shipped.

Another important factor in the context of client impact is when (and whether) clients obtained the list of revocations. Unfortunately, we are unable to answer this question given our data collection methodology (it would require instrumenting end-hosts to see when precisely their browsers and operating systems fetched CRLs or issued OCSP queries). Such a study is an interesting area of future work. However, there is one aspect of this problem to which we may be able to lend insight; it was recently reported that many browsers do not even bother to check certificates' CRLs, with the exception of extended validation (EV) certificates [7]. We next turn to an analysis of how these EV certificates are reissued and revoked in comparison to the entire corpus of certificates.

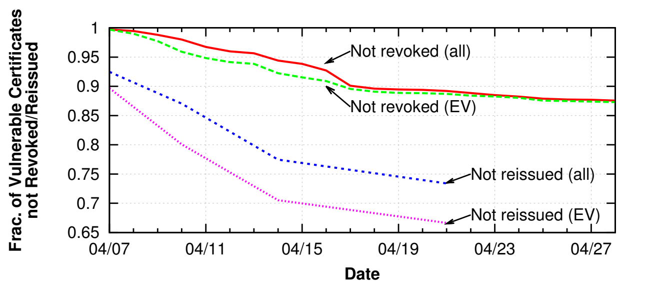

Figure 18: The rate at which vulnerable certificates were reissued and revoked after Heartbleed's announcement. (Note that the y-axis does not begin at zero.)

4.5 Extended Validation Certificates

Recall that one of the major roles of a CA is to validate the identity of the subjects for whom it issues certificates. *Extended Validation* (EV) certificates are a means by which CAs can express that this identity-verification process has followed a set of (presumably stringent) established criteria. EV certificates are standard X.509 certificates, and offer no additional security per se, but the rationale is that with a more thorough verification process by the CAs, these certificates can be more readily verified and trusted by users.[8] That said, there remains concern as to whether or not this trust is well-placed. We close this section by investigating the rate at which vulnerable EV certificates were revoked and reissued as compared to the entire aggregate of certificates.

Figure 18 shows the fraction of vulnerable certificates that have yet to be reissued or revoked over time. In this figure, the initial y values do not all start at 1.0 for reissues: this is because, with coarse granularity of our data, we cannot be certain whether some certificates were reissued immediately after the scan on April 7, 2014, immediately before the scan on April 10, 2014, or in between. We therefore provide the *most optimistic* possibility: if we know a certificate was reissued between days d and $d + k$, then we plot it as having been reissued on day d. The coarse granularity of the scans also explains why the reissue lines do not advance beyond April 21.

Regardless, one trend that remains clear is that sites are more proactive in reissuing new certificates than in revoking old ones. This contradicts prior assumptions that revocations and reissues occur simultaneously [8]. Indeed, it is not yet clear to us *why* a site would reissue a vulnerable certificate without revoking it, but these trends demonstrate that it is a common practice, even for those with EV certificates.

This figure shows a generally bleak view of how *thoroughly* sites revoke and reissue their certificates when necessary. Note that the y-axis begins at 0.65: three weeks after the revelation of Heartbleed, over 87% of all certificates we found to be vulnerable have yet to be revoked, and over 73% of them have yet to be reissued. Of those that *did* revoke their certificates, we find that the speed at which they did so matches that of earlier studies on the spread of patches [25, 27]: there is an exponential drop-off, followed by a gradual decline. Specifically, the "Not revoked (all)" line fits the

[8]Many browsers present EV certificates with a green box in the address bar, while non-EV certificates are often just represented with a gray lock icon.

curve $0.179e^{-0.073x} + 0.830$, while the "Not revoked (EV)" line fits the curve $0.144e^{-0.118x} + 0.859$.

Overall, EV certificates follow similar trends to the entire corpus, with a slightly faster and more thorough response. Interestingly, while EV certificates were revoked more quickly, their non-EV counterparts caught up within ten days; however, EV certificates were reissued both more quickly and more thoroughly. We expect that the underlying cause of this observation is a self-selection effect, i.e., security-conscious sites are more likely to seek out EV certificates in the first place. We doubt that the additional identity verification steps required to obtain an EV certificate play a large role in this (slightly) improved reaction to Heartbleed. Nonetheless, there are still many vulnerable EV certificates that have not been reissued two weeks after the event (67%) and that have not been revoked three weeks after (87%).

5. RELATED WORK

Our work lies at the intersection of two general areas of prior work: studies of how effectively administrators react to widely publicized vulnerabilities, and measurements of the TLS/SSL certificate ecosystem. To the best of our knowledge, we are the first to look specifically at how potentially compromised certificates are replaced and revoked.

Vulnerability patching. There have been several studies of how quickly and effectively administrators patch well-known software vulnerabilities. Rescorla measured the response to a 2002 buffer overflow vulnerability in OpenSSL [27], and Ramos investigated how the fraction of vulnerable systems changes after various security holes from 2000–2005 had been published [25]. Both of these studies found an exponential decrease in the fraction of vulnerable hosts shortly after public revelation of the vulnerability, followed by a gradual decline thereafter. Interestingly, in Rescorla's study, another sharp decline in the number of vulnerable hosts occurred after the release of the Slapper worm which exploited the buffer overflow.

Closely related to our study is that of Yilek et al., who measured the aftermath of a 2008 vulnerability in Debian's OpenSSL key generation that resulted in predictable RSA keys [35]. What makes this work particularly related to ours is that fixing the vulnerability required not only patching OpenSSL, but also reissuing new keys. They found that this process resulted in a gradual decline in the fraction of vulnerable hosts, as opposed to the sharp exponential decay when only patching the software is necessary. However, because their data collection only began several days after the vulnerability was released, the sharp decline may have occurred but gone unnoticed. Our data covers months leading up to and weeks after Heartbleed, allowing us more confidence in the initial drop-off of vulnerabilities.

Our work broadly builds on these prior studies in that we focus on a different, though equally important, aspect of the vulnerability fixing cycle: when potentially compromised certificates were not only replaced, but explicitly *revoked*. The connection between patching software, reissuing new certificates, and revoking old ones has, to the best of our knowledge, not been explicitly studied. Though it had been previously believed that revocations and reissues occur simultaneously [8], our results demonstrate that revocations are often offset in time, or simply never occur at all.

The certificate ecosystem. In focusing on vulnerability fixing as it pertains to certificates, our work is also related to recent studies of the certificate ecosystem at large. Holz et al. [17] performed passive and active measurements on HTTPS certificates from the Alexa Top-1M domains. Durumeric et al. [8] performed active measurements using ZMap [10] that yielded nearly 40× more certificates than prior studies [11,16,17]. Broadly, these studies exposed several grim properties of today's certificate ecosystem, including weaker key lengths than suggested by NIST [3], longer certificate chains than necessary, invalid subject names, and so on. Comparing these studies to one another, it appears that the Alexa Top-1M sites—though still far from perfect—do manage certificates more appropriately on average, with a slight weight to higher-ranked domains. Like Holz et al., our work focuses solely on the Alexa Top-1M; we expect that expanding to more domains would, as Durumeric et al. found [8], result in less effective certificate management, though this is an area of future work.

While these studies have shed considerable light on the certificate ecosystem (and found it to be surprisingly bleak), our study is the first to explicitly consider reissues and revocations, particularly in the wake of a widespread vulnerability. Durumeric et al. [8] briefly investigated certificate revocations, and found that a mere 2.5% of the certificates they encountered were ever revoked—of these, the majority gave no reason code. By using Heartbleed as a wide-scale correlated event, we complement this prior work by investigating which certificates *should* have been revoked, and *when* the revocations should have taken place. In the context of the certificate ecosystem, we believe this to be novel.

Heartbleed. The recent nature of the Heartbleed vulnerability means little scientific work has yet to come out studying the vulnerability itself and the community's reaction to it. The most closely related work—a study performed concurrently with our own—presents a comprehensive study of the breadth of the vulnerability, the clean-up, and surveys of administrators who failed to patch their servers [9]. Interestingly, the study leverages historic packet traces [19] to look for evidence of Heartbleed exploitation *before* the announcement and finds no evidence that the vulnerability was exploited beforehand. This study and our own are complementary—theirs briefly examines SSL certificate reissues and revocations, and the results of their analysis are in agreement with ours.

6. CONCLUDING DISCUSSION

In this paper, we study how SSL certificates are reissued and revoked in response to a widespread vulnerability, Heartbleed, that enabled undetectable key compromise. We conducted large-scale measurements and developed new methodologies and heuristics to determine how the most popular 1 million web sites reacted to this vulnerability in terms of certificate management, and how this impacts security for clients that use them.

We found that the vast majority of vulnerable certificates have not been reissued; further, of those domains that reissued certificates in response to Heartbleed, 60% do not revoke their vulnerable certificates—if they do not eventually become revoked, 20% of those certificates will remain valid (not expire) for two or more years. The ramifications of this findings are alarming: modern Web browsers will remain potentially vulnerable to malicious third parties using stolen keys to masquerade as a compromised site for a long time to come. We analyzed these trends with vulnerable EV certificates, as well, and have found that, while they exhibit better security practices, they still remain largely not reissued (67%) and not revoked (88%) even weeks after the vulnerability was made public.

To the best of our knowledge, our focused study on certificate reissues and revocations is the first of its kind. Our results are, in some ways, in line with previous studies on the rates at which administrators patched vulnerable software—for instance, revocation rates followed a sharp exponential drop-off shortly after the vulnerability was made public, and tapered off relatively soon thereafter. However, unlike with software patches, we find the vast majority of certificates have still not been reissued or revoked. These findings indicate quite simply that the current practices of certificate management are misaligned with what is necessary to ensure a secure PKI.

Surveying system administrators. To help better understand the reasons behind the lack of prompt certificate reissues and revocations, we informally surveyed a few systems administrators. We asked what steps they had taken in response to Heartbleed: did they patch, reissue, and revoke, and if not, then why not? We received seven responses. Most reported patching their systems, typically in direct response, but some relied on managed servers or automatic updates and therefore took no Heartbleed-specific steps. There was some variance in when patches were applied, due to a combination of scheduled reboots and delayed responses from some vendors, but the majority of patches were applied quickly.

For revoking and reissuing, however, we saw a wide spectrum of behavior. Few both revoked and reissued, but among them, they did so within 48 hours. Many neither revoked nor reissued; a common reason provided was that the vulnerable hosts were either not hosting sensitive data or were not running services that were deemed sensitive enough to warrant it. Along similar reasons, others reported having reissued the certificate but not revoking, explaining that the certificate is only for internal use. Finally, others reported that they did not perceive reissuing and revoking as important because they had patched quickly after the bug was publicly announced (recall, however, that the vulnerability was introduced over two years prior).

Our results from this small survey should be viewed anecdotally—a more extensive survey on certificate administration is an interesting area of future work—but they do shed light on some of the root causes of why revoking and reissuing are not on equal footing with patching. While administrators almost universally understand the importance of patching after a vulnerability, many do not appreciate or know about the importance of revoking and reissuing certificates with new keys. Of those administrators who do understand the importance, even some of them reported pushback from others who perceived the process as being overly complex. In sum, this points to the need for broader education on the treatment of certificates, and perhaps more assistance from CAs to help ensure that all the prescribed steps are taken.

Lessons learned. Our results suggest several changes to common PKI practices that may improve security in practice. First, the practices of low revocation rates and long expiration dates form a dangerous combination. Techniques that automate revocation would vastly reduce the period during which clients are vulnerable to malicious third parties. Similarly, setting reasonably short certificate expiration dates (as suggested by Topalovic et al. [34]) by default will significantly reduce the period during which vulnerable certificates are valid. Second, mechanisms that enable a simultaneous reissue-and-revoke for a certificate will make it less likely that invalid certificates are accepted by clients. Third, we have found that many domains, when they reissue a certificate, continue to offer the old, vulnerable certificate, as well. Given the large number of certificates and hosts using them per domain in our dataset, we believe administrators would benefit from tools that more easily track and validate the set of certificates they are using.

Future work. This paper is, we believe, the first step towards understanding the manual process of reissuing and revoking certificates in the wake of a vulnerability. Several interesting open problems remain. Because our data focuses on the server and CA side of the PKI ecosystem, we are unable to draw any direct conclusions as to what clients experience. A host-centered measurement study would, for instance, allow us to understand not only when revocations were added to CRLs, but when clients actually received the CRLs. Moreover, our study opens many questions as to *why* the certificate reissue and revocation processes are so extensively mismanaged. Our results reinforce previous findings that site popularity is correlated with good security practices, but even the highest ranked Alexa websites show relatively anemic rates of reissues and revocations. Understanding the root causes is an important step towards developing secure infrastructures that effectively incorporate (or mitigate) the end-user administrators.

Open source. Our analysis relied on both existing, public sources of data and those we collected ourselves. We make all of our data and our analysis code available to the research community at

`https://ssl-research.ccs.neu.edu`

Acknowledgments

We thank the anonymous reviewers and our shepherd, Jelena Mirkovic, for their helpful comments. We also thank Rapid7 for collecting the SSL certificate data, the authors of ZMap for collecting the Heartbleed vulnerability data, and for making it publicly available. Finally, we thank our survey respondents for their candid responses. This research was supported in part by NSF grants CNS-1054233, CNS-1319019, and CNS-1150177, and an Amazon Web Services in Education grant.

7. REFERENCES

[1] D. E. 3rd. Transport Layer Security (TLS) Extensions: Extension Definitions, Jan. 2011. IETF RFC-6066.

[2] Alexa Top 1 Million Domains. `http://s3.amazonaws.com/alexa-static/top-1m.csv.zip`.

[3] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. Recommendation for Key Management – Part 1: General (Revision 3), 2012. NIST Special Publication 800-57.

[4] Botan SSL Library. http://botan.randombit.net.

[5] CERT Vulnerability Note VU#720951: OpenSSL TLS heartbeat extension read overflow discloses sensitive information. http://www.kb.cert.org/vuls/id/720951.

[6] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. IETF RFC-5280, May 2008.

[7] R. Duncan. How certificate revocation (doesn't) work in practice, 2013. http://news.netcraft.com/archives/2013/05/13/how-certificate-revocation-doesnt-work-in-practice.html.

[8] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman. Analysis of the HTTPS certificate ecosystem. In *ACM Internet Measurement Conference (IMC)*, 2013.

[9] Z. Durumeric, J. Kasten, F. Li, J. Amann, J. Beekman, M. Payer, N. Weaver, J. A. Halderman, V. Paxson, and M. Bailey. The matter of Heartbleed. In *ACM Internet Measurement Conference (IMC)*, 2014.

[10] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *USENIX Security Symposium*, 2013.

[11] P. Eckersley and J. Burns. An observatory for the SSLiverse. In *Defcon 18*, 2010. https://www.eff.org/files/DefconSSLiverse.pdf.

[12] F. F. Elwailly, C. Gentry, and Z. Ramzan. QuasiModo: Efficient certificate validation and revocation. In *Public Key Cryptography (PKC)*, 2004.

[13] P. Evans. Heartbleed bug: RCMP asked Revenue Canada to delay news of SIN thefts, 2014. http://www.cbc.ca/news/business/heartbleed-bug-rcmp-asked-revenue-canada-to-delay-news-of-sin-thefts-1.2609192.

[14] Faketime library. http://www.code-wizards.com/projects/libfaketime/.

[15] B. Grubb. Heartbleed disclosure timeline: who knew what and when, 2014. http://www.smh.com.au/it-pro/security-it/heartbleed-disclosure-timeline-who-knew-what-and-when-20140415-zqurk.html.

[16] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman. Mining your Ps and Qs: Detection of widespread weak keys. In *USENIX Security Symposium*, 2012.

[17] R. Holz, L. Braun, N. Kammenhuber, and G. Carle. The SSL landscape – A thorough analysis of the X.509 PKI using active and passive measurements. In *ACM Internet Measurement Conference (IMC)*, 2011.

[18] Revocation doesn't work. https://www.imperialviolet.org/2011/03/18/revocation.html.

[19] S. Kornexl, V. Paxson, H. Dreger, A. Feldmann, and R. Sommer. Building a time machine for efficient recording and retrieval of high-volume network traffic. In *ACM Internet Measurement Conference (IMC)*, 2005.

[20] Mac OS X 10.9.2 Root Certificates. http://support.apple.com/kb/HT6005.

[21] S. Micali. NOVOMODO: Scalable certificate validation and simplified PKI management. In *PKI Research Workshop*, 2002.

[22] P. Mutton. Half a million widely trusted websites vulnerable to heartbleed bug, 2014. http://news.netcraft.com/archives/2014/04/08/half-a-million-widely-trusted-websites-vulnerable-to-heartbleed-bug.html.

[23] M. Naor and K. Nissim. Certificate revocation and certificate update. In *USENIX Security Symposium*, 1998.

[24] OpenSSL Project. https://www.openssl.org.

[25] T. Ramos. The laws of vulnerabilities. In *RSA Conference*, 2006. http://www.qualys.com/docs/Laws-Presentation.pdf.

[26] Rapid7 SSL Certificate Scans. https://scans.io/study/sonar.ssl.

[27] E. Rescorla. Security holes... Who cares? In *USENIX Security Symposium*, 2003.

[28] R. L. Rivest. Can we eliminate certificate revocation lists? In *Financial Cryptography (FC)*, 1998.

[29] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP, June 2013. IETF RFC-6960.

[30] R. Seggelmann, M. Tuexen, and M. Williams. Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension, Feb. 2012. IETF RFC-6520.

[31] N. Sullivan. The Heartbleed Aftermath: all CloudFlare certificates revoked and reissued, 2014. http://blog.cloudflare.com/the-heartbleed-aftermath-all-cloudflare-certificates-revoked-and-reissued.

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

[33] The GnuTLS Transport Layer Security Library. http://www.gnutls.org.

[34] E. Topalovic, B. Saeta, L.-S. Huang, C. Jackson, and D. Boneh. Toward short-lived certificates. In *Web 2.0 Security & Privacy (W2SP)*, 2012.

[35] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage. When private keys are public: Results from the 2008 Debian OpenSSL vulnerability. In *ACM Internet Measurement Conference (IMC)*, 2009.

[36] P. Zheng. Tradeoffs in certificate revocation schemes. In *ACM Computer Communication Review (CCR)*, 2013.

[37] ZMap Vulnerable Hosts. https://zmap.io/heartbleed/vulnerable.html.