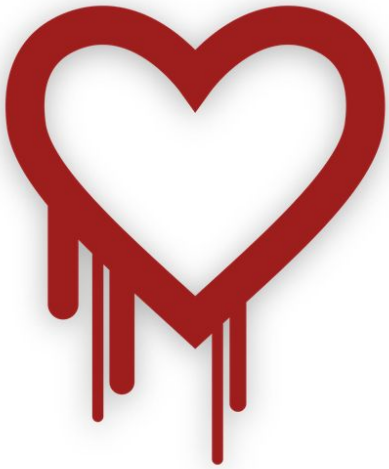
Heartbleed

By James, Fink, and Paule.



Overview

Heartbleed is a vulnerability in the OpenSSL heartbeat mechanism. It allows attackers to exploit the server, causing it to provide private memory which may include user passwords, private keys, and/or other sensitive data. Heartbleed was inadvertently created by an OpenSSL engineer and likely overworked graduate student, Robin Segglemann, at 11:59PM on New Year's Eve in 2011. It was unfortunately not caught during its code review and was merged into OpenSSL version 1.0.1. The error was a failure to add a bounds check on user input within the heartbeat extension.

The problem was discovered by Neel Mehta at Google and the Codenomicon security firm, independently, and was disclosed on April 1st, 2014. Several incidents relating to heartbleed have occurred. On April 8th, 2014 the Canada Revenue Agency had 900 social insurance numbers stolen during a 6 hour window via the vulnerability. Upon discovering the theft Canada Revenue Agency took their website offline and their engineers quickly patched the vulnerability by updating their server's OpenSSL version. Canada Revenue Agency provided anyone affected by the theft with credit monitoring services at no cost as well. Another notable attack was against Community Health Systems, the second largest for-profit U.S. hospital chain. They were attacked by Chinese hackers using the vulnerability, leading to the compromise of 4.5 million patient records about a week after the vulnerability was disclosed. The heartbleed vulnerability was exploited to retrieve private keys used to encrypt user credentials. Upon the credentials being decrypted, the attackers had full capability to enter user accounts containing private information.

The severity of the vulnerability is high with a CVSS impact rating of 2.9 and exploitability rating of 10.0. The impact rating is relatively low because it's primarily related to provider specific implementations. That being said, it's impact is significant in that the memory segments potentially exposed by exploitation are likely to contain private keys and other sensitive information that would enable attackers to perform more attacks once retrieved (as in the example of the attack on Community Health Systems). However, the ease of exploitation of the vulnerability makes it an attack vector that is widely available, thereby increasing the risk of the attack occurring in the wild significantly.

In response to the disclosure, on April 7th, 2014 the vulnerability was patched and OpenSSL consumers were asked to upgrade their OpenSSL packages and revoke/regenerate any keys previously created using OpenSSL as they may have been compromised during their exposure period. On April 11th 2014 Cloudflare created an open challenge for hackers to attempt to retrieve private keys from a server using the vulnerability. Later that day, several winners were named, signifying the relative severity of the vulnerability and it's ease of exploitation.

A few important terms related to heartbleed are:

Bounds Checking - The process of checking that a value is within specified boundaries before being used.

Buffer Over-read - A circumstance where more of a machine's buffer is read than intended.

Heartbeat Extension - An extension to TLS and DTLS to enable connections to be kept-alive even when not actively being used for data transfer.

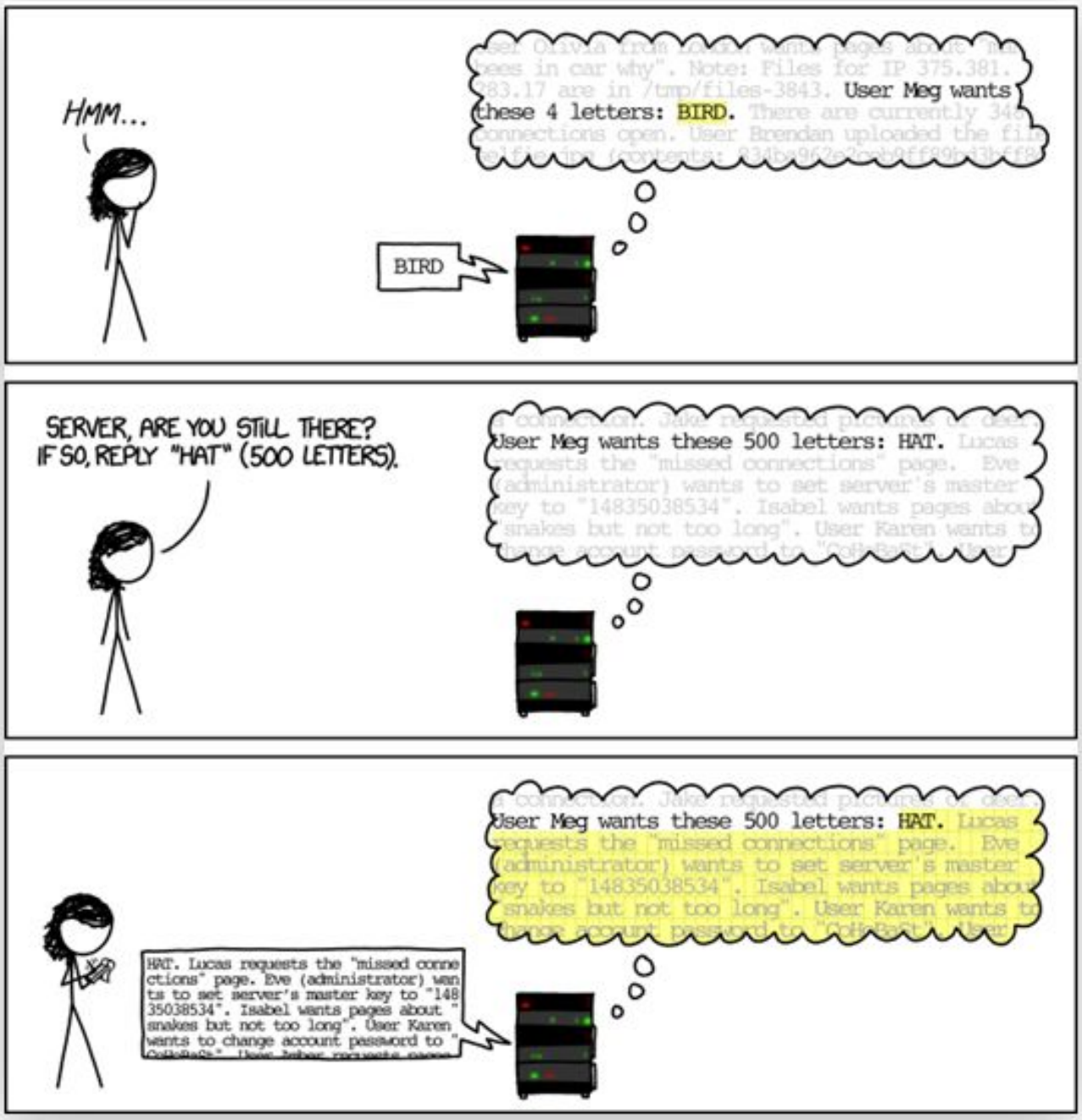


Figure A: This XKCD comic illustrates the problem introduced.
https://xkcd.com/1354/

Technical Expose

Heartbleed is a remote vulnerability that effects web connected software using OpenSSL v1.0.1-1.0.1f for TLS with the heartbeat extension enabled. It attacks a buffer over-read vulnerability in the OpenSSL implementation of the TLS heartbeat extension. It is not delivered as a payload, rather, it abuses the TLS heartbeat extension to remotely obtain private memory used by the server application. There is no delivery mechanism beyond a maliciously crafted heartbeat message.

There aren't any public any malware campaigns that make use of the vulnerability. However, the vulnerability provides an attack vector. So like phishing it provides a means through which an attacker may begin to deliver malware to users and systems. For example, if a server's private key is retrieved via the vulnerability, it is then possible to spoof the server and execute a MITM attack against users that believe they're visiting the real website. It may provide private keys capable of decrypting usernames and passwords, thereby enabling access of privileged areas within a system. Or if the private keys retrieved are used to decrypt legitimate TLS traffic, users' encrypted packets can be decrypted, making the use of TLS pointless. That being said, unrelated to the actual vulnerability, there is a spam phishing malware campaign, heartbleedbugremovaltool.exe, that targets users by claiming to be a heartbleed removal tool which is actually a keylogger. So due to the widespread and necessary communication of the vulnerability to those affected, less technical users are now more at risk as malicious actors have taken advantage of its publicity.

Millions of people have been affected by the vulnerability, indirectly. Several entities have utilized the vulnerability although given its widespread nature attribution to particular entities is challenging or impossible. This issue with attribution was confirmed by a honeypot study of heartbleed usage in the wild setup by researchers at the University of Michigan. In essence, a slight majority of the IP addresses that attacked the honeypot came from Chinese IP addresses, but there isn't any particular reason to believe the actors are themselves are located in China or are Chinese. In addition, the study lacked sufficient attacks to validate regional usage, given only a total of 41 attacks were performed all together.

What heartbleed does is provide up to 64KB of private memory in response to a malicious heartbeat request. The code that causes the over-read is memcpy(bp, pl, payload); in the extension program. bp points to a position in the server's memory, pl is the data the client sent with the heartbeat, and payload is the byte size of pl. The problem is payload may not contain the correct size of pl. When it is larger than pl additional memory (payload - pl) is returned in the response as well. The information in this response is dependent upon whatever the server had in the over-read space at the time of the request.

The vulnerability only exists when there is a network connection to the target device. If there is no network connection, no heartbeat can be received. The vulnerability doesn't cause

the vulnerable machine's filesystem to be modified and the vulnerability naturally doesn't hide itself.

Containment Strategy

17.5 percent of websites which used OpenSSL were affected at the time of disclosure. Over half a million (17.5% of SSL site) certificates were compromised. The vulnerability was widespread and potentially severe. The heartbleed vulnerability needs to be treated as a critical problem as it is expected to be utilized imminently due to its ease of exploitation.

The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to encrypt the traffic, the names and passwords of the users and the actual content. This allows attackers to eavesdrop on communications, steal data directly from the services and users and to impersonate services and users. Naturally the effect of the vulnerability is contained within the private memory of the software utilizing the vulnerable OpenSSL version. Removing the affected service or device from the public internet mitigates the chances of the vulnerability being exploited.

As long as the vulnerable version of OpenSSl is in use the machine can be abused therefore patching should be done immediately upon the discovery of the vulnerability if it exists.

Awareness Training

OpenSSL is known to be a software used by the web server to do encryption through the internet. The thing is that openSSL is generally secured but also a risky software used by most internet. The reason why is because it can not be protected by firewalls while in the web servers where sensitive information is collected at. Since 1998 openSSL has been running the internet with no issues upon till the heartbleed vulnerability has been discovered. Based on research reports the impact initially was ~17.5% of websites using TLS were affected, which totaled ~0.5 million domains. Of these, ~8K of the most popular websites at the time were vulnerable. By June 21st 2014 309K web servers remained vulnerable. By January 23rd 2017 180K internet-connected devices were vulnerable. By July 6th 2017 144K internet-connected devices remained vulnerable. By July 11th 2019 91K internet-connected devices remain vulnerable.

The financial cost at the level of all people affected was challenging to determine due to its long lifetime as a zero-day vulnerability and its possible use as an attack vector against various entities during this period, and after its disclosure. Depending on a specific provider's infrastructure, the costs of reissuing all their certificates would be in the range of ~$400K - $1M. Given this reissuing must be done by several providers the total cost is much higher. In total, costs to fix the heartbleed vulnerability globally added with the total cost on providers relating to

associated breaches is significant. Based on some projections for example, the cost of identity theft per person is ~$1.3K. Based on this, the single breach at Canada Revenue Agency would cost ~$1.3K x 900 victims = ~$1.17M (assuming all those affected have their identity stolen as a result of the breach).

Operating systems are not directly affected by Heartbleed, software running on the OS using the vulnerable OpenSSL version, making use of it for TLS, and which have the heartbeat extension enabled are at risk. When under attack via the vulnerability check all servers, devices, both internal and external-facing systems and applications for anything running OpenSSL 1.0.1 through 1.0.1f. Patching isn't automated and involves the participation of customers and users. Performing a risk assessment to determine what information may have been accessible during the attack and finding what needs to be done in relation is necessary as well, beyond technical measures to remediate the vulnerability.

We suggest that system admins approach the search for vulnerable applications by checking the versions of OpenSSL utilized and comparing them with the vulnerable versions of OpenSSL. Second, online scanners can be used to identify vulnerable servers. Since both library version checking and online scanning won't capture third party software that contains vulnerable OpenSSL libraries, it is important to contact any third party vendors utilized to determine if they are vulnerable and what their remediation timeline is. During that time it may be necessary to halt activity with the provider until their software is patched depending on the nature of the information communicated.

If found, the company should immediately remove the vulnerable device/software from public networks and patch the device and/or software with an updated version of OpenSSL. Keep customers in the loop on what has been found vulnerable and your company's remediation strategy is important, without notifying users their information is likely breached, they will not be able to resolve problems within their lives as a result of the breach. In addition to patching vulnerable OpenSSL libraries it's important to reissue certificates after patching as those created beforehand may have been breached. Changing passwords on any impacted systems should not be the first step to take because it will have no value until the affected servers are patched and relevant certificates reissued. However, once servers have been patched and certificates replaced, proceeding with password resets is ideal.

So, how much of a risk does it remain and what is the future outlook on the vulnerability? As stated earlier, as of July 11th 2019 ~91K devices remain vulnerable. While this number is significantly lower than the initial total, the severity of an attack against individual devices varies significantly so it likely still leaves many users at significant risk. Based on the fact that ~129K devices were patched in the first 2.5 years since it's disclosure, and ~90K were patched in the following 2.5 years, we suspect the vulnerability may continue to exist in the wild for another ~5 years or longer if the rate of devices patched continues to decrease year to year.

Resources

https://www.theguardian.com/technology/2014/apr/18/heartbleed-bug-will-cost-millions

https://www.cnet.com/news/heartbleed-bug-what-you-need-to-know-faq/

https://gigaom.com/2014/04/08/heres-everything-you-need-to-know-about-the-heartbleed-web-security-flaw/

https://news.softpedia.com/news/Chinese-Hackers-Are-Scanning-the-Web-for-Websites-Vulnerable-to-Heartbleed-Attacks-437943.shtml

https://gizmodo.com/how-heartbleed-works-the-code-behind-the-internets-se-1561341209

https://www.cnet.com/news/heartbleed-bug-what-you-need-to-know-faq/

https://www.dailydot.com/debug/heartbleed-bug-robin-seggelmann/

https://en.wikipedia.org/wiki/Heartbleed

https://www.webopedia.com/TERM/H/heartbleed-bug.html

https://www.cbc.ca/news/business/heartbleed-bug-rcmp-asked-revenue-canada-to-delay-news-of-sin-thefts-1.2609192

https://time.com/3148773/report-devastating-heartbleed-flaw-was-used-in-hospital-hack

https://nvd.nist.gov/vuln/detail/CVE-2014-0160#vulnCurrentDescriptionTitle

https://blog.cloudflare.com/the-hard-costs-of-heartbleed/

https://www.csid.com/2016/09/real-cost-identity-theft/

https://blog.cloudflare.com/the-results-of-the-cloudflare-challenge/

https://github.com/indutny/heartbleed

https://tools.ietf.org/html/rfc6520

http://heartbleed.com/