The Heartbleed Bug

The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication security and privacy over the Internet for applications such as web, email, instant messaging (IM) and some virtual private networks (VPNs).

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.

What leaks in practice?

We have tested some of our own services from attacker's perspective. We attacked ourselves from outside, without leaving a trace. Without using any privileged information or credentials we were able steal from ourselves the secret keys used for our X.509 certificates, user names and passwords, instant messages, emails and business critical documents and communication.

How to stop the leak?

As long as the vulnerable version of OpenSSL is in use it can be abused. Fixed OpenSSL (https://www.openssl.org/news/secadv/20140407.txt) has been released and now it has to be deployed. Operating system vendors and distribution, appliance vendors, independent software vendors have to adopt the fix and notify their users. Service providers and users have to install the fix as it becomes available for the operating systems, networked appliances and software they use.

Q&A

What is the CVE-2014-0160?

CVE-2014-0160 is the official reference to this bug. CVE (Common Vulnerabilities and Exposures) is the Standard for Information Security Vulnerability Names maintained by MITRE (http://cve.mitre.org/). Due to co-incident discovery a duplicate CVE, CVE-2014-0346, which was assigned to us, should not be used, since others independently went public with the CVE-2014-0160 identifier.

Why it is called the Heartbleed Bug?

Bug is in the OpenSSL's implementation of the TLS/DTLS (transport layer security protocols) heartbeat extension (RFC6520). When it is exploited it leads to the leak of memory contents from the server to the client and from the client to the server.

What makes the Heartbleed Bug unique?

Bugs in single software or library come and go and are fixed by new versions. However this bug has left large amount of private keys and other secrets exposed to the Internet. Considering the long exposure, ease of exploitation and attacks leaving no trace this exposure should be taken seriously.

Is this a design flaw in SSL/TLS protocol specification?

No. This is implementation problem, i.e. programming mistake in popular OpenSSL library that provides cryptographic services such as SSL/TLS to the applications and services.

What is being leaked?

Encryption is used to protect secrets that may harm your privacy or security if they leak. In order to coordinate recovery from this bug we have classified the compromised secrets to four categories: 1) primary key material, 2) secondary key material and 3) protected content and 4) collateral.

What is leaked primary key material and how to recover?

These are the crown jewels, the encryption keys themselves. Leaked secret keys allow the attacker to decrypt any past and future traffic to the protected services and to impersonate the service at will. Any protection given by the encryption and the signatures in the X.509 certificates can be bypassed. Recovery from this leak requires patching the vulnerability, revocation of the compromised keys and reissuing and redistributing new keys. Even doing all this will still leave any traffic intercepted by the attacker in the past still vulnerable to decryption. All this has to be done by the owners of the services.

What is leaked secondary key material and how to recover?

These are for example the user credentials (user names and passwords) used in the vulnerable services. Recovery from this leak requires owners of the service first to restore trust to the service according to steps described above. After this users can start changing their passwords and possible encryption keys according to the instructions from the owners of the services that have been compromised. All session keys and session cookies should be invalidated and considered compromised.

What is leaked protected content and how to recover?

This is the actual content handled by the vulnerable services. It may be personal or financial details, private communication such as emails or instant messages, documents or anything seen worth protecting by encryption. Only owners of the services will be able to estimate the likelihood what has been leaked and they should notify their users accordingly. Most important thing is to restore trust to the primary and secondary key material as described above. Only this enables safe use of the compromised services in the future.

What is leaked collateral and how to recover?

Leaked collateral are other details that have been exposed to the attacker in the leaked memory content. These may contain technical details such as memory addresses and security measures such as canaries used to protect against overflow attacks. These have only contemporary value and will lose their value to the attacker when OpenSSL has been upgraded to a fixed version.

Recovery sounds laborious, is there a short cut?

After seeing what we saw by "attacking" ourselves, with ease, we decided to take this very seriously. We have gone laboriously through patching our own critical services and are dealing with possible compromise of our primary and secondary key material. All this just in case we were not first ones to discover this and this could have been exploited in the wild already.

How revocation and reissuing of certificates works in practice?

If you are a service provider you have signed your certificates with a Certificate Authority (CA). You need to check your CA how compromised keys can be revoked and new certificate reissued for the new keys. Some CAs do this for free, some may take a fee.

Am I affected by the bug?

You are likely to be affected either directly or indirectly. OpenSSL is the most popular open source cryptographic library and TLS (transport layer security) implementation used to encrypt traffic on the Internet. Your popular social site, your company's site, commerce site, hobby site, site you install software from or even sites run by your government might be using vulnerable OpenSSL. Many of online services use TLS to both to identify themselves to you and to protect your privacy and transactions. You might have networked appliances with logins secured by this buggy implementation of the TLS. Furthermore you might have client side software on your computer that could expose the data from your computer if you connect to compromised services.

How widespread is this?

The most notable software using OpenSSL are the open source web servers like Apache and nginx. The combined market share of just those two out of the active sites on the Internet was over 66% according to Netcraft's April 2014 Web Server Survey (https://news.netcraft.com/archives/2014/04/02/april-2014-web-server-survey.html). Furthermore OpenSSL is used to protect for example email servers (SMTP, POP and IMAP protocols), chat servers (XMPP protocol), virtual private networks (SSL VPNs), network appliances and wide variety of client side software. Fortunately many large consumer sites are saved by their conservative choice of SSL/TLS termination equipment and software. Ironically smaller and more progressive services or those who have upgraded to latest and best encryption will be affected most. Furthermore OpenSSL is very popular in client software and somewhat popular in networked appliances which have most inertia in getting updates.

What versions of the OpenSSL are affected?

Status of different versions:

- OpenSSL 1.0.1 through 1.0.1f (inclusive) are vulnerable
- OpenSSL 1.0.1g is NOT vulnerable
- OpenSSL 1.0.0 branch is NOT vulnerable
- OpenSSL 0.9.8 branch is NOT vulnerable

Bug was introduced to OpenSSL in December 2011 and has been out in the wild since OpenSSL release 1.0.1 on 14th of March 2012. OpenSSL 1.0.1g released on 7th of April 2014 fixes the bug.

How common are the vulnerable OpenSSL versions?

The vulnerable versions have been out there for over two years now and they have been rapidly adopted by modern operating systems. A major contributing factor has been that TLS versions 1.1 and 1.2 came available with the first vulnerable OpenSSL version (1.0.1) and security community has been pushing the TLS 1.2 due to earlier attacks against TLS (such as the BEAST (https://en.wikipedia.org/wiki/Transport_Layer_Security#BEAST_attack)).

How about operating systems?

Some operating system distributions that have shipped with potentially vulnerable OpenSSL version:

- Debian Wheezy (stable), OpenSSL 1.0.1e-2+deb7u4
- Ubuntu 12.04.4 LTS, OpenSSL 1.0.1-4ubuntu5.11
- CentOS 6.5, OpenSSL 1.0.1e-15
- Fedora 18, OpenSSL 1.0.1e-4
- OpenBSD 5.3 (OpenSSL 1.0.1c 10 May 2012) and 5.4 (OpenSSL 1.0.1c 10 May 2012)
- FreeBSD 10.0 - OpenSSL 1.0.1e 11 Feb 2013
- NetBSD 5.0.2 (OpenSSL 1.0.1e)
- OpenSUSE 12.2 (OpenSSL 1.0.1c)

Operating system distribution with versions that are not vulnerable:

- Debian Squeeze (oldstable), OpenSSL 0.9.8o-4squeeze14
- SUSE Linux Enterprise Server
- FreeBSD 8.4 - OpenSSL 0.9.8y 5 Feb 2013
- FreeBSD 9.2 - OpenSSL 0.9.8y 5 Feb 2013
- FreeBSD 10.0p1 - OpenSSL 1.0.1g (At 8 Apr 18:27:46 2014 UTC)
- FreeBSD Ports - OpenSSL 1.0.1g (At 7 Apr 21:46:40 2014 UTC)

How can OpenSSL be fixed?

Even though the actual code fix may appear trivial, OpenSSL team is the expert in fixing it properly so fixed version 1.0.1g or newer should be used. If this is not possible software developers can recompile OpenSSL with the handshake removed from the code by compile time option `-DOPENSSL_NO_HEARTBEATS`.

Should heartbeat be removed to aid in detection of vulnerable services?

Recovery from this bug might have benefitted if the new version of the OpenSSL would both have fixed the bug and disabled heartbeat temporarily until some future version. Majority, if not almost all, of TLS implementations that responded to the heartbeat request at the time of discovery were vulnerable versions of OpenSSL. If only vulnerable versions of OpenSSL would have continued to respond to the heartbeat for next few months then large scale coordinated response to reach owners of vulnerable services would become more feasible. However, swift response by the Internet community in developing online and standalone detection tools quickly surpassed the need for removing heartbeat altogether.

Can I detect if someone has exploited this against me?

Exploitation of this bug does not leave any trace of anything abnormal happening to the logs.

Can IDS/IPS detect or block this attack?

Although the heartbeat can appear in different phases of the connection setup, intrusion detection and prevention systems (IDS/IPS) rules to detect heartbeat have been developed. Due to encryption differentiating between legitimate use and attack cannot be based on the content of the request, but the attack may be detected by comparing the size of the request against the size of the reply. This implies that IDS/IPS can be programmed to detect the attack but not to block it unless heartbeat requests are blocked altogether.

Has this been abused in the wild?

We don't know. Security community should deploy TLS/DTLS honeypots that entrap attackers and to alert about exploitation attempts.

Can attacker access only 64k of the memory?

There is no total of 64 kilobytes limitation to the attack, that limit applies only to a single heartbeat. Attacker can either keep reconnecting or during an active TLS connection keep requesting arbitrary number of 64 kilobyte chunks of memory content until enough secrets are revealed.

Is this a MITM bug like Apple's goto fail bug was?

No, this does not require a man in the middle attack (MITM). Attacker can directly contact the vulnerable service or attack any user connecting to a malicious service. However in addition to direct threat the theft of the key material allows man in the middle attackers to impersonate compromised services.

Does TLS client certificate authentication mitigate this?

No, heartbeat request can be sent and is replied to during the handshake phase of the protocol. This occurs prior to client certificate authentication.

Does OpenSSL's FIPS mode mitigate this?

No, OpenSSL Federal Information Processing Standard (FIPS) mode has no effect on the vulnerable heartbeat functionality.

Does Perfect Forward Secrecy (PFS) mitigate this?

Use of Perfect Forward Secrecy (PFS), which is unfortunately rare but powerful, should protect past communications from retrospective decryption. Please see https://twitter.com/ivanristic/status/453280081897467905 (https://twitter.com/ivanristic/status/453280081897467905) how leaked tickets may affect this.

Can heartbeat extension be disabled during the TLS handshake?

No, vulnerable heartbeat extension code is activated regardless of the results of the handshake phase negotiations. Only way to protect yourself is to upgrade to fixed version of OpenSSL or to recompile OpenSSL with the handshake removed from the code.

Who found the Heartbleed Bug?

This bug was independently discovered by a team of security engineers (Riku, Antti and Matti) at Codenomicon (https://www.synopsys.com/software-integrity/security-testing/fuzz-testing.html) and Neel Mehta of Google Security, who first reported it to the OpenSSL team. Codenomicon team found heartbleed bug while improving the SafeGuard feature in Codenomicon's Defensics security testing tools and reported this bug to the NCSC-FI for vulnerability coordination and reporting to OpenSSL team.

What is the Defensics SafeGuard?

The SafeGuard feature of the Codenomicon's Defensics security testtools automatically tests the target system for weaknesses that compromise the integrity, privacy or safety. The SafeGuard is systematic solution to expose failed cryptographic certificate checks, privacy leaks or authentication bypass weaknesses that have exposed the Internet users to man in the middle attacks and eavesdropping. In addition to the Heartbleed bug the new Defensics TLS Safeguard feature can detect for instance the exploitable security flaw in widely used GnuTLS open source software implementing SSL/TLS functionality and the "goto fail;" (https://gotofail.com/) bug in Apple's TLS/SSL implementation that was patched in February 2014.

Who coordinates response to this vulnerability?

Immediately after our discovery of the bug on 3rd of April 2014, NCSC-FI (https://www.kyberturvallisuuskeskus.fi/en) took up the task of verifying it, analyzing it further and reaching out to the authors of OpenSSL, software, operating system and appliance vendors, which were potentially affected. However, this vulnerability had been found and details released independently by others before this work was completed. Vendors should be notifying their users and service providers. Internet service providers should be notifying their end users where and when potential action is required.

Is there a bright side to all this?

For those service providers who are affected this is a good opportunity to upgrade security strength of the secret keys used. A lot of software gets updates which otherwise would have not been urgent. Although this is painful for the security community, we can rest assured that infrastructure of the cyber criminals and their secrets have been exposed as well.

What can be done to prevent this from happening in future?

The security community, we included, must learn to find these inevitable human mistakes sooner. Please support the development effort of software you trust your privacy to. Donate money to the OpenSSL project. (https://www.openssl.org/support/donations.html)

Where to find more information?

This Q&A was published as a follow-up to the OpenSSL advisory, since this vulnerability became public on 7th of April 2014. The OpenSSL project has made a statement at https://www.openssl.org/news/secadv_20140407.txt (https://www.openssl.org/news/secadv/20140407.txt). Individual vendors of operating system distributions, affected owners of Internet services, software packages and appliance vendors may issue their own advisories.

References

- CVE-2014-0160
- NCSC-FI case# 788210
- OpenSSL Security Advisory (https://www.openssl.org/news/secadv/20140407.txt) (published 7th of April 2014, ~17:30 UTC)
- CloudFlare: Staying ahead of OpenSSL vulnerabilities (https://blog.cloudflare.com/staying-ahead-of-openssl-vulnerabilities) (published 7th of April 2014, ~18:00 UTC)
- heartbleed.com (http://heartbleed.com/) (published 7th of April 2014, ~19:00 UTC)
- Ubuntu / Security Notice USN-2165-1 (https://www.ubuntu.com/usn/usn-2165-1/)
- FreeBSD / SA-14:06.openssl (https://www.freebsd.org/security/advisories/FreeBSD-SA-14:06.openssl.asc)
- FreshPorts / openssl 1.0.1_10 (https://www.freshports.org/security/openssl/)

- RedHat / RHSA-2014:0376-1 (https://access.redhat.com/errata/RHSA-2014:0376)
- CentOS / CESA-2014:0376 (https://lists.centos.org/pipermail/centos-announce/2014-April/020249.html)
- Fedora / Status on CVE-2014-0160 (https://lists.fedoraproject.org/pipermail/announce/2014-April/003205.html)
- CERT/CC (USA) (https://www.kb.cert.org/vuls/id/720951/)
- CERT.at (Austria) (https://www.cert.at/warnings/all/20140408.html)
- CIRCL (Luxembourg) (http://www.circl.lu/pub/tr-21/)
- CERT-FR (France) (https://www.cert.ssi.gouv.fr/)
- JPCERT/CC (Japan) (https://www.jpcert.or.jp/english/at/2014/at140013.html)
- CERT-SE (Sweden) (https://www.cert.se/2014/04/ny-sarbarhet-i-openssl-1-0-1)
- CNCERT/CC (People's Republic of China) (https://www.cert.org.cn/publish/main/9/2014/20140410091311360563426/20140410091311360563426_.html)
- Public Safety Canada (https://www.publicsafety.gc.ca/cnt/rsrcs/cybr-ctr/2014/av14-017-en.aspx)
- LITNET CERT (Lithuania) (https://cert.litnet.lt/2015/05/heartbleed-openssl-pazeidziamumas-3/)
- UNAM-CERT (Mexico) (https://www.cert.org.mx/historico_cert/boletin/index.html@vulne=6487)
- SingCERT (Singapore) (https://www.csa.gov.sg/singcert/news/advisories-alerts/updates-for-openssl-heartbleed-vulnerabilty-for-end-users)
- Q-CERT (Qatar) (http://www.qcert.org/sites/default/files/public/alert_details_9042014_heartbleed.pdf)

Page updated 2014-04-29 07:05 UTC.

(https://www.synopsys.com/software-integrity/security-testing/fuzz-testing.html)