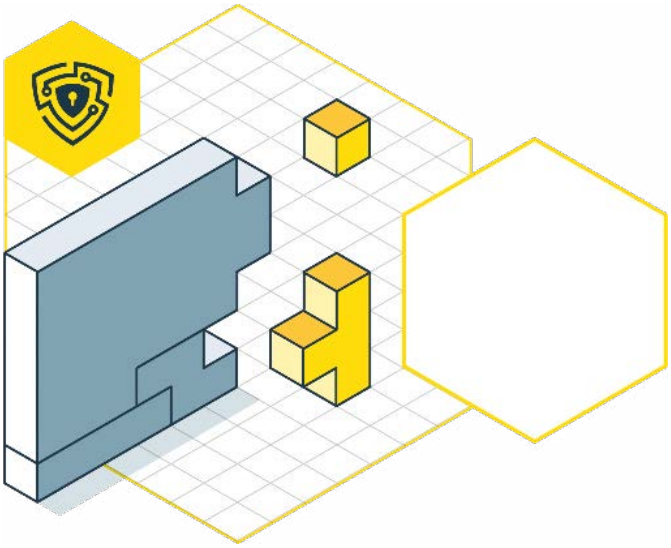
CRASHTEST **SECURITY**

GUIDE FOR
SSL HEARTBLEED VULNERABILITY PREVENTION

WHAT ARE THE STEPS TO KEEP YOUR WEB APP OR API SAFE FROM SUCH VULNERABILITY

GUIDE TO
SSL HEARTBLEED VULNERABILITY PREVENTION

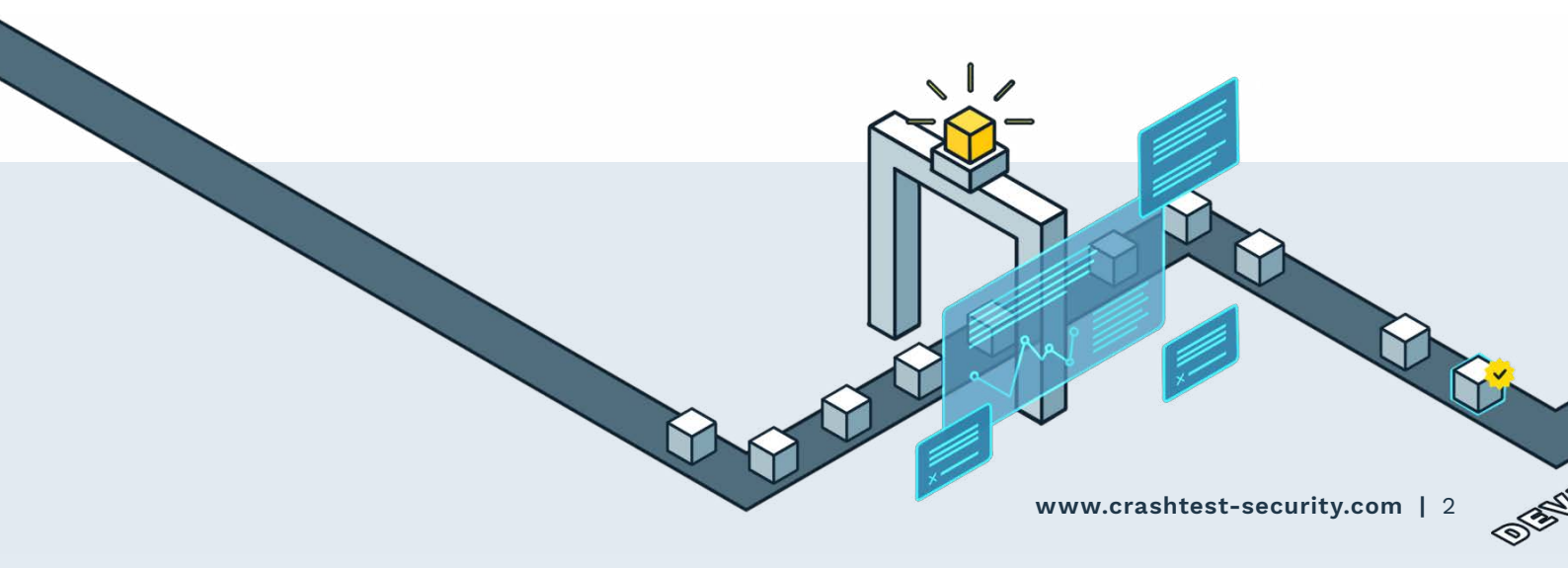
Table of Contents



INTRODUCTION TO THIS GUIDE

OpenSSL is an open-source cryptographic toolkit that offers a wide range of cryptographic algorithms and protocols. OpenSSL leverages the secure sockets layer (SSL) protocol to help maintain a secure connection among services by using encryption algorithms to prevent malicious actors from accessing data in transit. Despite its numerous benefits, incorrect implementation of the OpenSSL leads to a Heartbleed vulnerability that allows attackers to steal information normally protected by SSL cryptographic functions.

This guide discusses the heartbleed security flaw, its impacts, and prevention measures.



WHAT IS THE HEARTBLEED BUG?

The Heartbleed vulnerability is a security issue found in OpenSSL's implementation of the Heartbeat Extension for TLS protocol. Heartbeat is an echo functionality built into SSL that keeps a secure connection alive to ensure the server and the client correctly handle data encryption and decryption. A heartbeat request contains user data and random padding bits, where the receiving entity responds by echoing the data in the initial request with different padding. In vulnerable OpenSSL versions, an attacker can read up to 64 KB of the padding memory buffer, allowing them to expose encrypted content.

Some operating systems ship with OpenSSL versions affected by this critical vulnerability, including:

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

RECENT IMPACTS OF THE HEARTBLEED BUG

Instead of being a vulnerability with the SSL protocol, the Heartbleed bug is caused by the approach of implementing the Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) protocols by OpenSSL. Security researchers have recorded real-world exploits of the Heartbleed vulnerability. However, there have been claims that these were tests by the National Security Agency to determine the extent of the vulnerability.

When the vulnerability was published, most enterprises relying on vulnerable OpenSSL versions rushed to implement a fix for it and other heartbleed-like vulnerabilities, costing thousands of dollars. Wary of the impacts of an attack, some platforms restrained users from accessing their services until they could implement functional improvements over complex structures of the OpenSSL libraries, resulting in a loss of revenue.

Some enterprises also failed to implement the OpenSSL Heartbleed fix before attackers crafted successful exploits. These include:

1. The **Canadian Revenue Agency** reported a data breach that exposed users' Social Insurance Numbers (SINs) and linked it to the Heartbleed bug. The attackers accessed and deleted 900 SINs using compromised memory allocations, which could have been potentially uploaded to a malicious database or used for credit card fraud and identity theft.

2. In another instance, the Heartbleed bug was also blamed for a data breach on the **Community Health Systems** platform. Attackers successfully breached security measures to protect the platform's private networks and transferred non-medical patient data to external serv

HEARTBLEED VULNERABILITY - SEVERITY LEVEL

Uncovered by security experts of Codenomicon in 2014, the Heartbleed bug was traced to an inherent programming flaw within the OpenSSL cryptographic library of versions 1.0.1 through 1.0.1f. Heartbleed is a severe security issue since it allows threat actors to access restricted data from target memory allocations by triggering memory buffer over-reads in complex structures.

The bug's severity is that it can be exploited remotely without authentication or authorization. Additionally, the exploit can access memory areas containing sensitive data such as private keys, passwords, and other personal information.

Sensitive data exposed through the Heartbleed bug is categorized into:

- **Primary key material** - These are the secret keys used to encrypt sensitive information in traffic. If attackers obtain primary key material, they can decrypt any future traffic to the protected online services and potentially keep impersonating vulnerable services. Depending on the transport protocols, the attacker can intercept all traffic between the internet and the firm's private networks. While security teams can recover compromised services by patching the security flaw and reissuing new keys, traffic intercepted by the malicious actor in the past may not be fully recovered.

- **Secondary key material** - Secondary key material comprises the usernames, passwords, and other credentials used to access accounts in vulnerable services. In instances where attackers obtain these credentials, they can compromise user accounts, allowing them the privileges and permissions of legitimate users. Depending on the roles assigned to the affected user accounts, leakage of secondary material can be used for a wide variety of attacks, including credit card fraud, code injection, and a compromise of the entire private network, among others. Once a secondary key material has been exposed, the application can be recovered by revoking compromised credentials and restoring trust in the primary key material.

- **Protected Content -** Protected content represents sensitive data being handled by vulnerable services. These include encrypted communications such as instant messaging and emails, personal details, business-critical information, restricted files & documents, and personal and financial details. When attackers gain access to protected content, they can perform different exploits depending on the content exposed. To ensure the safe use of vulnerable services and data, it is advisable to restore trust to the primary and secondary keys.

- **Collateral -** These represent technical details of memory allocations that attackers can use for advanced attacks. Collateral details include standard memory allocator configurations, settings of memory caching systems, data memory allocations, security settings, and other functional improvements built to protect encrypted data. Since collateral material only has contemporary value, attackers lose their leverage when a vulnerable OpenSSL version is upgraded to a secure version.

The vulnerability is listed in the **Common Vulnerabilities and Exposures (CVE) database as CVE-2014-0160** and is also associated with one **Common Weakness Enumeration entry: Buffer Over-read (CWE 126)**. This weakness is caused when the application reads from a buffer using access mechanisms that reference memory allocations beyond the targeted buffer. A typical attack mechanism uses the Heartbleed bug to read data from out-of-bounds memory to get collateral values (such as memory buffer addresses) for advanced attacks.

HOW TO IDENTIFY SSL HEARTBLEED WITH CRASHTEST SECURITY?

The Crashtest Security Suite implements an automated vulnerability scanning and penetration testing mechanism that helps security researchers detect and remediate heartbleed-like vulnerabilities. The suite includes a dedicated Heartbleed Tester that helps security teams check their web servers for SSL misconfigurations that may lead to the Heartbleed bug. Since Heartbleed attacks leave no traces in event logs, the Heartbleed tester helps administer an essential combined practice of advanced behavior detection, identification, and mitigation solution. **Crashtest Security's Heartbleed Tester** also checks the OpenSSL library for known attack vectors and provides actionable reports with no false positives to enable quick remediation.

Crashtest Security also offers an SSL/TLS scanner to identify software vulnerabilities in the Transport Layer Security protocol to restrict memory buffer overreads. Besides the Heartbleed vulnerability, this scanner also detects other critical heartbleed-like vulnerabilities, such as POODLE, BEAST, LUCKY13, and BREACH.

HEARTBLEED ATTACK PREVENTION TECHNIQUES

Some security measures to prevent Heartbleed attacks include:

OPENSSL HEARTBLEED FIX

The first step to preventing Heartbleed attacks is upgrading vulnerable OpenSSL versions with the latest version of its cryptographic library. The fix was released in OpenSSL version 1.0.1g and has been included in all subsequent versions. The code fix for vulnerable OpenSSL versions ensures that the SSL protocol's heartbeat request is not empty and the request remains alive for the designated period.

Although there is no evidence that the flaw has been exploited in the wild, it is recommended that users of vulnerable systems adopt appropriate measures to protect their data. This includes patching systems to fix the bug, revoking and reissuing certificates for affected servers, and changing passwords for services that use weak encryption methods.

INPUT VALIDATION

Attackers typically target improper input validation errors that help leak memory contents to craft malicious heartbeat requests. To prevent this, developers and security teams should assume all user input is malicious. When performing input validation, security teams should adopt a white list approach. This approach helps admit only those requests that strictly conform to pre-designed specifications while rejecting everything else or transforming it into an acceptable input. When creating the whitelist, security experts should consider all relevant attributes, including request length, input type, all acceptable values, syntax, invalid inputs, and conformance to business-critical processes.

THROTTLING AND RESOURCE LIMITING

When using the OpenSSL cryptographic library, developers should clearly state the minimum, and maximum thresholds for memory buffer reads. When setting these limits, teams should also factor in the application's behavior once the heartbleed request approaches the limits.

Throttling is a measure that restricts the type and amount of memory content a user can access. Since attackers typically use multiple Heartbleed requests to leak data sets larger than 64 KB, tracking the requests received from particular users and blocking them once requests reach a certain threshold is important. A recommended practice is combining throttling mechanisms with access control models to identify the source of a memory over-read exploit.

PERFECT FORWARD SECRECY (PFS)

Perfect Forward Secrecy is a security mechanism that helps improve encryption by leveraging temporary key exchanges between a server and the client. Users are issued a unique session key when they log in. This key is separate from the cryptographic key, is private, and only lasts during the user's session. When an attacker has leveraged the Heartbleed bug to compromise one of the user's primary keys, the PFS mechanism helps keep the content secure and encrypted. A successful compromise also requires the session key. In other instances where cryptographic and session keys are compromised, only the contents of the specific session are potentially exploited while keeping other session details secure.

BEST PRACTICES IN PREVENTING HEARTBLEED SECURITY BUGS

Best practices to prevent Heartbleed vulnerability include:

ENFORCE OPEN-SOURCE SECURITY

Developers and security experts should perform Software Composition Analysis (SCA) to detect the presence of the Heartbleed vulnerability on any open-source components used in their application. Continuous open-source security scanning helps detect and identify unpatched Heartbleed vulnerabilities for remediation. One option is to use a free, open-source vulnerability scanner like OWASP's Dependency Checker that can be run locally on your development machine or server. Another option is to use a commercial vulnerability scanner, such as Crashtest Security, that helps with a comprehensive scan based on OWASP benchmarks and offers enterprise support.

Once you've identified vulnerable components, it is recommended to update them to the latest, stable version at the earliest. If you're using an open-source component that's no longer being maintained, you'll need to find a replacement component that is still supported.

REVOKE AND REPLACE EXISTING CERTIFICATES

In systems that use vulnerable versions of the OpenSSL library, it is essential to stop the bleeding before implementing any remediation measures. A recommended approach is creating a new key pair and certificate-signing request for all TLS/SSL certificates.

A common approach to replacing existing vulnerable services involves the following steps:

1. Check whether your website or service is affected by Heartbleed. You'll need to generate new SSL/TLS certificates if it is.
2. Revoke your existing certificate using the Certificate Authority (CA) that issued it.
3. Generate a new certificate using a different CA. Use strong encryption and follow all best practices for securing your certificates.
4. Install the new certificate on your website or service.
5. Test everything to make sure it's working correctly. In systems that use vulnerable versions of the OpenSSL library, it is essential to stop the bleeding before implementing any remediation measures. A recommended approach is creating a new key pair and certificate-signing request for all TLS/SSL certificates.

ENFORCE SERVER-SIDE INPUT VALIDATION

When performing input validation, the service should never trust input supplied by an external system. As external input validation may lack the ability to evaluate some attributes used in the private network's white list, the web server should evaluate a request on its own to determine whether it includes invalid inputs.

PREVENT HEARTBLEED-LIKE VULNERABILITIES WITH CRASHTEST SECURITY

Crashtest Security offers a suite of various security scanners to help security researchers detect and remediate software vulnerabilities. The platform's **TLS/SSL scanner** and **Heartbleed Tester** enable comprehensive security audits by automatically detecting known OpenSSL attack vectors within application frameworks. The platform also helps implement an automated security scanning mechanism to detect other heartbleed-like vulnerabilities with extremely low false positives.

To know more about how Crashtest Security can help prevent Heartbleed bug exploits, try a **free, 14-day demo** here.

Start 2-Week Trial for Free

PREVENT SSRF ATTACKS WITH CRASHTEST SECURITY

Crashtest Security helps administer an automatic scanning and testing framework to prevent server-side request forgery vulnerabilities and other modern web security risks. The platform continuously benchmarks applications against the OWASP Top 10 vulnerabilities to help mitigate critical security risks through proactive detection, identification, and remediation.

Crashtest Security seamlessly integrates security testing into development workflows to ensure threats are detected and remediated since the early stages of the SDLC. With its quick security assessment and actionable security reports, cross-functional teams can identify security blind spots and remediate threats faster.

To know more about how Crashtest Security can help eliminate SSRF vulnerabilities before they are exploited in production, **try a free, 14-day demo** here.

Start 2-Week Trial for Free

CRASHTEST **SECURITY**

WWW.CRASHTEST-SECURITY.COM