International Journal For Technological Research In Engineering
Volume 4, Issue 9, May-2017 *ISSN (Online): 2347 –4718*

OpenSSL Heartbleed vulnerability & prediction of severity of exploitation posed by some of the common types of vulnerabilities, based on Common Vulnerability Scoring System (CVSS), using Naive Bayes classification algorithm.

Mehak Bashir
M.Tech. (CSE), RN Engineering College, MDU, Rohtak, India.

***Abstract*—The Open Secure Sockets Layer (OpenSSL) provides secure tribune for transactions, such as online shopping, banking, emails & social media etc., that take place over/across the internet. Heartbleed sways numerous things, encompassing web servers, routers, mobile apps and VPNs (Virtual Private Network). It has been estimated that 60 percent of secure web sites that are operating on OpenSSL are affected. In addition, Heartbleed cannot be traced. The data that has been exploited is colossal and includes usernames, passwords, bank account and credit card numbers, documents in online cloud storage. Not only has all of this user data been directly compromised, but what is more serious, the private keys of the servers operating on the vulnerable versions of OpenSSL were more or less certainly exploited. The proposed research describes 'OpenSSL Heartbleed' vulnerability and also proposes a methodology that explains the severity of exploitation posed by some common types of vulnerabilities, based on Common Vulnerability Scoring System (CVSS), using Naive Bayes classification algorithm.**

Keywords—OpenSSL; Heartbleed bug; Vulnerability; CVSS Parameters; CVE numbers; Naive Bayes classification; Frequency table; Probability.

I. INTRODUCTION

The OpenSSL is an open source implementation of the Secure Sockets Layer (SSL) and the Transport Layer Security (TLS) [6]. This protocol is used in two-thirds (66%) of all websites to avert hackers from snipping sensitive information [8]. The OpenSSL protocol works by authenticating the server to the client and client to server through the use of digital certificates signed by a trusted third party. The OpenSSL protocol is however exposed to vulnerabilities [1], [3] either directly or indirectly. This can be discerned, how the trusted third parties, who authenticate the identities of transacting individual, have been exposed to continuous attacks/threats [6]. Various other vulnerabilities have been discovered within the OpenSSL protocol [2] and the most eminent has been the Heartbleed bug.

The name 'Heartbleed' itself demystifies the vulnerability – 'Heart' of the Heartbleed is taken from Heartbeat protocol and 'bleed' stands for data leakage [7]. Thus Heartbleed means data leakage in the Heartbeat protocol implementation, specifically the OpenSSL implementation of the protocol.

Naive Bayes is a kind of classifier which makes use of the Bayes Theorem. It predicts membership probabilities for each class such as the probability that given record or data point belongs to a particular class. The class with the highest probability is considered as the most likely class. This is also known as Maximum A Posteriori (MAP) [8].

II. LITERATURE SURVEY

In April 7, 2014, The Heartbleed Bug was independently discovered by a crew of security engineers (Riku, Antti, and Matti) at Codenomicon and Neel Mehta of Google Security, who premier reported it to the OpenSSL team. The security engineers did not have any notion of the vulnerability until they ascertained heartbleed bug, whilst they were improving the Safeguard features. They reported this bug to the NCSC-FI (National Cyber Security Center –Finland) for vulnerability coordination and to the OpenSSL team as well [4].

In addition, Bloomberg in year 2014 accused the U.S National Security Agency (NSA) of being acquainted about the Heartbleed Bug for the last two years. Albeit, the report enunciated that the NSA was using it to gain information instead of revealing it to the OpenSSL developer. After the NSA nixing to opine to report of being acquainted about the Heartbleed Bug, NSA also refuted that they were aware of Heartbleed Bug until the vulnerability was made public by the private security engineering of Google. Overall, the questions remain about whether anyone from the NSA or U.S government might have leveraged the code for their benefits before it was published to the public.

The Heartbleed Bug is not a virus, it's not a worm or a malicious code, and it has nothing to do with the Man-in-the-Middle, but it's a simple programming mistake. The programmer Robin Seggelmann, a 31 year old from Germany, submitted the code. The purpose of the software was to enable a function called "Heartbeat" in OpenSSL. This software package was to be employed by nearly half of all web servers to implement the connections between clients and servers. "In one of the new features, unfortunately, I missed validating a variable containing a length" (Seggelmann, 2012). In addition, the code went undetected by several code reviewers and everyone else for over two years. Stephen Solis-Reyes 19 year-old from Canada was nabbed for exploiting the Heartbleed Bug to attack the website of the Canada Revenue Agency. As result, of the attack, Mr. Solis-Reyes had stolen 900 social insurances numbers (Elsevier, 2014). Ivan Ristic, director of engineering at Qualys, has avowed that the percentage of websites vulnerable to the flaw of heartbleed had dropped from 25 percent since the bug was discovered. "Assistant Research Scientist Dave Levin and Assistant Professor of Electrical and Computer Engineering Tudor Dumitras were part of a team that analyzed the most popular websites in the United States-more than one million sites were examined-to better understand the extent to which systems administrators followed specific protocols to fix the problem"(NewsRx, 2014) [5] .

III. HEARTBLEED EXPLANATION

A. *How the heartbeat works*

The heartbeat extension protocol consists of two message types: HeartbeatRequest message and HeartbeatResponse message. When a receiver receives a HeartbeatRequest message, the receiver should transmit back an exact copy of the received message in the HeartbeatResponse message. The sender validates that the HeartbeatResponse message is same as what was originally sent. If it is same, the connection is kept alive. If the response does not contain the same message, the HeartbeatRequest message is retransmitted for a specified number of retransmissions.

B. *Data leakage as a result of Heartbleed*

There is a bug in the implementation of the Heartbeat reply to the received Heartbeat request message. Heartbeat reply copies the received payload to the Heartbeat response message to verify that the secured connection is still active, without examining if the payload length is same as the length of the request payload data.

If the heartbeat request payload length field is set to a value greater than the actual payload, it would result in a return of the payload followed by whatever contents are currently contained in active memory buffer, beyond the end of the payload. In heartbeat request the payload length can be set to a maximum value of 65535 bytes [2]. Therefore the bug in the OpenSSL heartbeat response code could copy as much as 65535 bytes from the machine's memory and send it to the requestor. This bug is illustrated below in "Fig. 1: Memory Leak [7]."

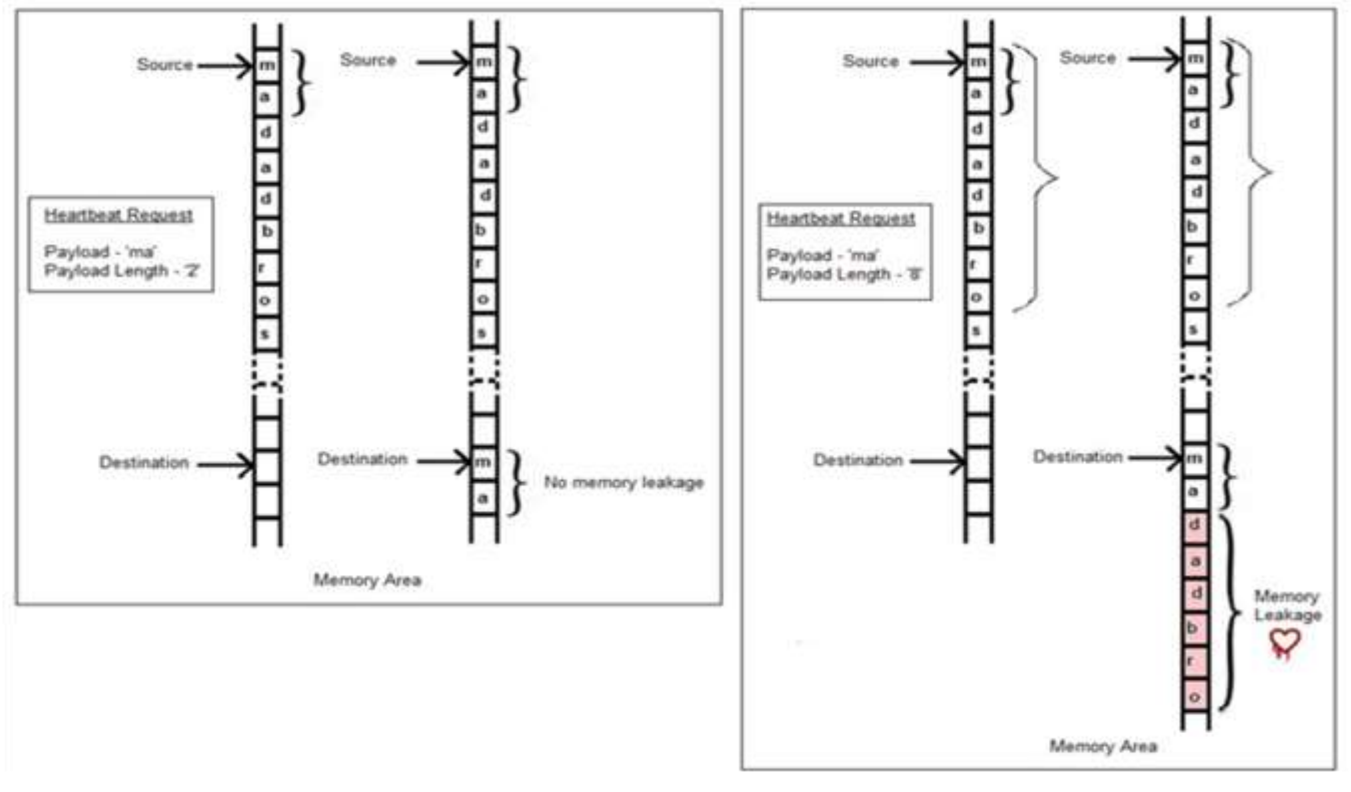

Fig. 1: Memory Leak

IV. Common Vulnerability Scoring System (CVSS)

The data used in this section comes from the National Vulnerability Database (NVD), which includes Information for all Common Vulnerabilities and Exposures (CVEs). Each CVE comes with some Common Vulnerability Scoring System (CVSS) metrics and parameters [9], which are tabulated in Table 1.

V. Naive Bayes Classifier

Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems. This methodology is easiest to understand when described using binary or categorical input values.

It is called naive Bayes or idiot Bayes because the calculation of the probabilities for each hypothesis is simplified to make their calculation tractable. Rather than attempting to calculate the values of each attribute value P (d1, d2, d3|h), they are assumed to be conditionally independent given the target value and calculated as P (d1|h) * P (d2|H) and so on [8].

A. *Algorithm for predicting severity/threat Of exploitation posed by common vulnerabilities using Naive Bayes approach*

- Convert the data set into a frequency table.
- Create Likelihood table by finding the probabilities, like probability of High threat of exploitation is (4/7) = 0.57 and probability of Low threat of exploitation is (3/7) = 0.43.
- Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

B. *Frequency table for some common vulnerabilities based on CVSS(Version2) parameters for predicting severity/threat of exploitation , using Naive Bayes Model*

The values for CVSS (Version2) parameters: CVSS Score, Access Vector, Access, Complexity, Authentication, Confidentiality, Integrity and Availability for some common types of vulnerabilities such as PhpMyAdmin Reflected cross- Site Scripting Vulnerability (CVE-2013-1937), MySQL Stored SQL Injection (CVE-2013-0375), SSL v3 POODLE Vulnerability (CVE_2014-3568), VMWare Guest to Host Escape Vulnerability (CVE-2012-1516), Apache Tomcat XML Parser Vulnerability (CVE-2009-0783), Cisco IOS Arbitrary Command Execution Vulnerability (CVE-2012-0384), Apple iWork Denial of Service Vulnerability (CVE-2015-1098), OpenSSL Heartbleed Vulnerability (CVE-2014-0160), GNU Bourne-Again Shell(Bash) 'ShellShock' Vulnerability (CVE-2014-6271), DNS Kaminsky Bug (CVE-2008-1447), Sophos Login ScreenByPass Vulnerability (CVE-2014-2005), Sophos Login ScreenByPass Vulnerability (CVE-2014-2005) etc. are tabulated in table 2.

TABLE 1 CVSS (version 2) Base Metrics, with definitions from Mell et al. (2007).

Parameter	**Values**	**Description**
CVSS Score	0-10 [Low (0.1 -3.9), Medium (4.0 – 6.9), High (7.0 – 8.9), Critical (9.0 – 10.0)]	This value is calculated based on the next six parameters, with a formula (Mell et al., 2007).
Access Vector	Local Adjacent Network	The access vector (AV) determines how vulnerability can be exploited. A local attack requires physical access to the computer or a shell account. Vulnerability with Network access is also called remotely exploitable.
Access Complexity	Low Medium High	The access complexity (AC) classifies the difficulty to exploit the vulnerability.
Authentication	None Single Multiple	The authentication (Au) categorizes the number of times that an attacker must authenticate to a target to exploit it, but does not measure the difficulty of the authentication process itself.
Confidentiality	None Partial Complete	The confidentiality (C) metric assorts the impact of the confiden-tiality, and amount of information access and disclosure. This may include partial or full access to file systems and/or database tables.
Integrity	None Partial Complete	The integrity (I) metric categorizes the impact on the integrity of the exploited system. For example, if the remote attack is able to partially or fully modify information in the exploited system.
Availability	None Partial Complete	The availability (A) metric categorizes the impact on the availability of the target system. Attacks that consume network bandwidth, processor cycles, memory or any other resources affect the availability of a system.

TABLE 2 Frequency table for some common vulnerabilities using CVSS (version 2) Base Metrics.

Vulnerability	**CVSS V2 Base score**	**Access Vector**	**Access Complexity**	**Authentic ation**	**Confidentiality**	**Integrity Impact**	**Availability Impact**	**Severity/ Threat of Exploitation**
PhpMyAdmin Reflected cross- Site Scripting Vulnerability (CVE-2013-1937)	Medium	Network	Medium	None	None	Partial	None	Low
MySQL Stored SQL Injection (CVE-2013-0375)	Medium	Network	Low	Single	Partial	Partial	None	High
SSL v3 POODLE Vulnerability (CVE_2014-3568)	Medium	Network	Medium	None	Partial	None	None	Low
VMWare Guest to Host Escape Vulnerability (CVE-2012-1516)	Critical	Network	Low	Single	Complete	Complete	Complete	High
Apache Tomcat XML Parser Vulnerability (CVE-2009-0783)	Medium	Local	Low	None	Partial	Partial	Partial	High
Cisco IOS Arbitrary Command Execution Vulnerability (CVE-2012-0384)	High	Network	Medium	Single	Complete	Complete	Complete	High
Apple iWork Denial of Service Vulnerability (CVE-2015-1098)	Medium	Network	Medium	None	Partial	Partial	Partial	Low
OpenSSL Heartbleed Vulnerability (CVE-2014-0160)	Medium	Network	Low	None	Partial	None	None	High
GNU Bourne-Again Shell(Bash) ‘ShellShock’ Vulnerability (CVE-2014-6271)	Critical	Network	Low	None	Complete	Complete	Complete	High
DNS Kaminsky Bug (CVE-2008-1447)	Medium	Network	Low	None	None	Partial	None	Low

TABLE 2 (continued)

Vulnerability	**CVSS V2 Base score**	**Access Vector**	**Access Complexity**	**Authentic ation**	**Confidentiality**	**Integrity Impact**	**Availability Impact**	**Severity/ Threat of Exploitation**
Joomla Directory Traversal Vulnerability (CVE-2010-0467)	Medium	Network	Low	None	Partial	None	None	Low
Cisco Access Control ByPass Vulnerability (CVE-2012-1342)	Medium	Network	Low	None	None	Partial	None	Low
Juniper Proxy ARP Denial of Service Vulnerability (CVE-2013-6014)	Medium	Adjacent	Low	None	None	Complete	None	High
DokuWiki Reflected Cross-Site Scripting Attack (CVE-2014-9253)	Medium	Network	Medium	None	None	Partial	None	Low
Adobe Acrobat Buffer Overflow Vulnerability (CVE-2009-0658)	Critical	Network	Medium	None	Complete	Complete	Complete	High
Microsoft Windows Bluetooth Remote Code Execution Vulnerability (CVE-2011-1265)	High	Network	Low	None	Complete	Complete	Complete	High
Apple ios Security control Bypass vulnerability (CVE-2014-2019)	Medium	Local	Low	None	None	Complete	None	High
SearchBlox Cross-Site Request Forgery Vulnerability (CVE-2015-0970)	Medium	Network	Medium	None	Partial	Partial	Partial	Low
SSL/TLS MITM Vulnerability (CVE-2014-0224)	Medium	Network	Medium	None	Partial	Partial	Partial	Low
Google Chrome ByPass Vulnerability (CVE-2012-5376)	Critical	Network	Low	None	Complete	Complete	Complete	High

C. Likelihood Table for finding the probabilities(P) Of various CVSS (version 2) parameters

The likelihood table for finding the probabilities of various CVSS parameters : CVSS Score, Access Vector, Access, Complexity, Authentication, Confidentiality, Integrity and Availability, as defined in table 1, using the data presented in above frequency table (table 2) is depicted in table 3.

TABLE 3 Likelihood table for calculation of probabilities of CVSS (version 2) parameters

Severity/Threat of Exploitation	
P (High) = 12/21 = 4/7	P (Low) = 9/ 21 = 3/7
CVSS V2 Base score	
P (Low/ High) = 0/12 = 0	P(Low/ Low) = 0/9 = 0
P (Medium/ High) = 6/12 =1/2	P(Medium/ Low) = 9/9 = 1
P (High/ High) = 2/12 = 1/6	P(High/ Low) = 0/9 = 0
P (Critical/ High) = 4/12 = 1/3	P(Critical/ Low) = 0/9 = 0
Access Vector (AV)	
P (Local/ High) = 3/12 = 1/4	P (Local/ Low) = 0/9 = 0
P (Adjacent/ High)= 1/12	P (Adjacent/ Low) = 0/9 = 0
P (network/ High) = 8/12 = 2/3	P (Network/ Low) = 9/9 = 1
Access Complexity (AC)	
P (Low/ High) = 9/12 = 3/4	P (Low/ Low) = 3/9 = 1/3
P (Medium/ High)= 3/12 = 1/4	P (Medium/ Low) = 6/9 = 2/3
P (High/ High) = 0/12 = 0	P (High/ Low) = 0/9 = 0
Authentication (Au)	
P (None/ High) = 9/12 = 3/4	P (None/ Low) = 9/9 = 1
P (Single/ High)= 3/12 = 1/4	P (Single/ Low) = 0/9 = 0
P (Multiple/ High) = 0/12 = 0	P (Multiple/ Low) = 0/9 = 0
Confidentiality Impact (C)	
P (None/ High) = 2/12 = 1/6	P (None/ Low) = 4/9
P (Partial/ High)= 3/12 = 1/4	P (Partial/ Low) = 5/9
P (Complete/ High) = 7/12	P (Complete/ Low) = 0/9 = 0
Integrity Impact (I)	
P (None/ High) = 1/12	P (None/ Low) = 2/9
P (Partial/ High)= 2/12 = 1/6	P (Partial/ Low) = 7/9
P (Complete/ High) = 9/12 = 3/4	P (Complete/ Low) = 0/9 = 0
Availability Impact (A)	
P (None/ High) = 4/12 = 1/3	P (None/ Low) = 6/9 = 2/3
P (Partial/ High)= 1/12	P (Partial/ Low) = 3/9 = 1/3
P (Complete/ High) = 7/12	P (Complete/ Low) = 0/9 = 0

D. Using Naive Bayes equation to calculate the posterior probability for a sample class of Vulnerability, to predict its severity of exploitation

Let A be a sample vulnerability with CVSS parameters as:
<Medium, Local, Low, None, Partial, Partial, Partial>

The posterior probability of sample class A, for given set of CVSS parameters, is calculated from table 3 as:

P (A/High) × P (High)
= P (Medium/High) × P (Local/High) ×
P (Low/High) × P (None/High) ×
P (Partial/High) × P (Partial/High) ×
P (Partial/High) × P (High)

P (A/High) × P (High)
= (1/2) × (1/4) × (3/4) × (3/4) ×
(1/4) × (1/6) × (1/12) × (4/7)

P (A/High) × P (High) = 36/258048

P (A/High) × P (High) = 0.0001395089

Now we will calculate P (A/Low) × P (Low) as:

P (A/Low) × P (Low)
= P (Medium/Low) × P (Local/Low) ×
P (Low/Low) × P (None/Low) ×
P (Partial/Low) × P (Partial/Low) ×
P (Partial/Low) × P (Low)

P (A/Low) × P (Low)
= (9/9) × (0) × (1/3) × (1) × (5/9) × (7/9) ×
(1/3) × (3/7)

P (A/Low) × P (Low) = 0

Now the highest posterior probability is calculated to be:

MAX {P (A/High) × P (High), P (A/Low) ×
P (Low)} = MAX {0.0001395089, 0}

MAX {P (A/High) × P (High), P (A/Low) ×
P (Low)} = 0.0001395089

Since {P (A/High) × P (high)} is evaluated to be greater than {P (A/Low) × *P* (Low)}, hence the sample vulnerability class A with the CVSS parameters as:
< Medium, Local, Low, None, Partial, Partial, Partial> is predicted to pose high threat of exploitation and thus should quickly be reported for immediate remediation, to prevent the hackers from stealing the valuable data.

VI. CONCLUSION & RECOMMANDATIONS

All Heartbleed vulnerable systems should incontinently be upgraded to OpenSSL 1.0.1g. Version1.0.1g has bounds checking included to prevent buffer over- read [5].

Naive Bayes Classification enables us to prioritize vulnerabilities for remediation. The type of vulnerabilities which are classified as highly exploitable by the proposed methodology ,can be easily exploited with minimum efforts by the hackers, therefore the particular vulnerability needs headlong attention and should be remediated & fixed as early as possible, to prevent the exploitation of any kind.

REFERENCES

[1]. Yogesh Joshi, Debabrata Das, Subir Saha, "Mitigating Man in the Middle Attack over Secure Sockets Layer," IEEE, pp 1-5, 2009.
[2]. Eman Salem Alashwali , "Cryptographic Vulnerabilities in Real-Life Web Servers," In Proceedings of the The 3rd International Conference on Communication and Information Technology(ICCIT-2013):Digital Information Management and Security Beirut, pp 6-11, 2013.
[3]. Krishna Kant and Ravishankar Iyer, Prasant Mohapatra, "Architectural Impact of Secure Socket Layer on Internet Servers: A Retrospect," IEEE pp 25-26, 2012.
[4]. Neal Leavitt "Internet Security under Attack: The Undermining of Digital Certificates," IEEE Computer Society, pp 17-20, 2011.
[5]. University of Maryland; cybersecurity experts discover lapses in heartbleed bug fix. (2014). NewsRx Health & Science, Retrieved from http://search.proquest.com. jproxy.lib.ecu.edu/docview/1626397458?accountid=10639
[6]. OpenSSL Team" OpenSSL Project," openssl.org, 2014 [Online]. Available: https://www.openssl.org/ [Accessed: June. 12, 2014]
[7]. CODENOMICON "The Heartbleed Bug," heartbleed.com , 29 April 2014 [Online]. Available: http://heartbleed.com/. [Accessed: June. 12, 2014].
[8]. Rish, Irina (2001). An empirical study of the naive Bayes classifier(PDF). IJCAI Workshop on Empirical Methods in AI.
[9]. "Common Vulnerability Scoring System, V2 Development Update". First.org, Inc. Retrieved November 13, 2015.