

top security threats review

AN INDUSFACE REPORT

am I on this list ?



CONTENTS

03	Introduction	
04	Heartbleed	
	What is Heartbleed Vulnerability?	06
	How Critical is Heartbleed?	06
	What were the fixes?	07
	Effect of Heartbleed: Then & Now	07
	Initial Heartbleed victims	08
	Is Heartbleed all cured?	09
11	Poodle	
	Can you detect Poodle attacks on your network?	12
15	BASH	
18	Ghost Vulnerability	
	CVE-2015-0235 Basics	19
	Risk Analysis	20
	Affected Operating Systems	20
	Mitigation	20
21	FREAK	
	Where did the FREAK come from	22
	How is the FREAK vulnerability exploited?	23
	How can you ensure security?	24
26	SQL Injection	
	How to prevent injections like SQL	28
	Last Year: Wall Street Journal Got SQL-ed	29
	More on wOrm, the hacker who loves big data	30
31	DDoS	
	Last Year: DDoS Attacks hit the World Cup!	32
33	XSS	
	Stored XSS	34
	Reflected XSS	34
	DOM-based XSS	34
	Last Year: XSS Haunted Amazon and Word Press	35
	How to protect yourself from XSS?	36

INTRODUCTION



When you think of cyber vandalism and exploitation, think of water. It takes the path of least resistance and flows through the little gaps. In the internet world, these little spaces and loopholes that allow exploitation to flow through your system are called vulnerabilities.

And a lot of vulnerabilities notoriously registered themselves in the Internet hall of fame, continuing to haunt giant organizations. These threats are all over the news. When so much is happening around, we are sure you would have missed out on some of the key stories...and since we understand your need to stay updated on everything important, we bring to you an eBook with all the key details from Indusface, at one place. We hope that you stay updated and continue to overcome challenges successfully with us.

all that you knew,
and a lot more

about Heartbleed:

The event that
shook the
World Wide Web





On April 7th, a major vulnerability called Heartbleed was discovered in OpenSSL, the most prevalent software used for encryption and other purposes on the web and the internet. SSL, known as secure socket layer, is the preferred protocol used for encryption. OpenSSL is its most common and open-source implementation.

Websites that use encryption, payment gateways, VPNs, apps — including mobile apps, all use SSL and a large majority of them use OpenSSL. The two most common webservers Apache and Nginx, that comprises of more than 60% of web servers on the internet use OpenSSL when they use the https (that is the encrypted version) version of http. Most operating systems use OpenSSL for various modules, so these modules are also affected.

What is Heartbleed Vulnerability?

Heartbleed vulnerability was discovered by three researchers — Neel Mehta from Google and two others. Heartbleed allows a malicious user to steal sensitive information such as private keys, passwords etc. The vulnerability is present in a module of OpenSSL called TLS heartbeat extension which is used to generate heart beat messages. Hence the name Heartbleed for this vulnerability. This heartbeat handshake is usually done during the negotiation time of the SSL protocol and much before https takes over, in case SSL is used under https. Thus, the vulnerability is not present in layer 7 but rather at layer 4.



How critical is Heartbleed?

Heartbleed is a critical vulnerability. To get into a bit more detail, the Heartbleed vulnerability allows a malicious user using a client to get 64K of memory from the server. Now, this memory can potentially contain sensitive data such as private keys. Once one gets the private keys, the server can be impersonated. Thus Heartbleed needs to be fixed or taken care of immediately.

While Heartbleed existed in the OpenSSL software since about two years back, it was discovered only recently. The disclosure was made public on April 7th along with a version of OpenSSL (1.0.1.g) that has the fix. It could not be found if Heartbleed had been exploited. Further, if exploited, Heartbleed leaves no trace in the logs etc. unless one logs every SSL/TLS transaction which is hardly practical. This made it more difficult to find if it had been exploited in the past, but any server running a vulnerable version of OpenSSL was at risk.

What were the fixes?

OpenSSL had released a version of OpenSSL called 1.0.1g which would fix the vulnerability. This was released on 7th April. So, the best option was to upgrade from a vulnerable version to 1.0.1g. The other option was to recompile from source the version of OpenSSL which you were using

but with the flag `-DOPENSSL_NO_HEARTBEATS`. This would disable heartbeats and so circumvent the vulnerability.

Further, it was important to regenerate all keys generated from a vulnerable version. In case one was using an OS such as Ubuntu, as versions of Ubuntu from 12.04 were vulnerable. Ubuntu had made an upgrade version (1.0.1-4ubuntu5.12) available at

<http://askubuntu.com/questions/444702/how-to-patch-theheartbleed-bug-cve-2014-0160-in-openssl>

Effect of Heartbleed: **then and now**

The news of Heartbleed spread like a wildfire. Within days, multiple articles and websites came up with “The Heartbleed Hit List”- A list which comprised of most of the websites which were compromised and whose users were requested to change their password immediately. Big and popular names like Google, Yahoo, and Dropbox were listed. There was a great confusion on changing the passwords- or not changing them.

In June, six more bugs were found in OpenSSL and OpenSSL came up with a security advisory detailing seven vulnerabilities with their fixed versions. While most of the vulnerabilities could lead to denial of service attacks and arbitrary code execution, one of the vulnerabilities (CVE-2014-024) allowed a hacker to launch a MITM (man in the middle attack) and snoop on unencrypted data. The hacker could thus look at sensitive data in the clear, beating OpenSSL’s encryption. The impact for this could be enormous.

Heartbleed OpenSSL pulled with itself into limelight all the digital certificate-issuing authorities in the world. Emphasis on the need for support from them became overwhelming and none of them could shy away from that. The massive impact of this can be seen in the below graph, which shows revocations of a huge number of certificates within a few days. The greater good that came from Heartbleed is the recent announcement of the Core Infrastructure Initiative (CII), which funds open-source projects that are in the critical path for core computing functions. Many large technology firms came in support to financially support the key

open-source initiatives, signaling the beginning of the time where critical open source projects will be adequately backed up by the biggies.

CCI will be supporting Network Time Protocol, OpenSSH, and OpenSSL and other key projects, such as the Open Crypto Audit Project.

Initial Heartbleed Victims?

Many high profile websites came forwards as confirmed targets of Heartbleed attacks:

According to a Forbes article, Yahoo was exposed for about 24 hours where other sites like the Canadian Revenue Agency immediately took their website down while they worked on patching and remediation. The CRA commissioner Andrew Treusch later announced that 900 taxpayers' details including social security numbers, which could be used to gain access to government benefits or perform identity theft, were exposed by an attacker using Heartbleed. Shortly following the announcement the Royal Canadian Mounted Police announced the arrest of Mr. Solis-Reyes who was accused of stealing the 900 records and faced more than 16 charges last year including hacking & intercepting computer functions.

Another victim of Heartbleed, Mumsnet, also announced to its users that it had been attacked. Mumsnet, is a popular British parenting website with 1.5 million users. As per Forbes, the e-mail to users stated "On Thursday 10 April we at Mumsnet HQ became aware of the bug and immediately ran tests to see if the Mumsnet servers were vulnerable. As soon as it became apparent that we were, we applied the fix to close the OpenSSL security hole... However, it seems that users' data was accessed prior to our applying this fix". Mumsnet posted an article outlining how the attacker was able to log in as the founder of Mumsnet, Justine Roberts after using Heartbleed to steal her username and password.

Is Heartbleed all cured?

Marco Ostini, an Information security analyst working at Australian Computer Emergency Response Team (AusCERT) at the University of Queensland, recently stated that the OpenSSL vulnerabilities are very close to being universal, and are not restricted to server-side computing. As a result, they are affecting almost every operating system, many of which are yet to receive the patches for OpenSSL vulnerabilities.

With time, IT infrastructure has become complex, and the security of them more so. Trying to manage them on your own can be tricky. It's best to rely on a professional for the same. Many times, in the effort to save money, organizations decide to rely on their IT teams for security problems, which can work in some cases, but complex issues require more expertise. It's best to nip the problem at the bud stage, rather than when it becomes a weed and becomes too difficult to manage.

The same issue has been faced by IT teams in the case of Heartbleed. The Heartbleed bug was a programming mistake which went un-noticed for many years. Being used by a very large section of the world's internet, the remedial action to be taken was also of massive proportion. Securing the infected system not only required the updating of the software but also obtaining new "master keys" to re-establish their corporate electronic identity. In many cases, the users needed to be requested to change their passwords.

When six more bugs were found in June, more remedial efforts were required by the IT teams. We won't be surprised if the cost of dealing with Heartbleed, globally, has already touched millions. And the money would be well spent if Heartbleed was fixed for good, but unfortunately the problem still persists.

There is a long list of products affected by OpenSSL Heartbleed, encompassing almost any IT product or service imaginable. One example is the 4.1.1 Version of Android's Jelly Bean OS, which is still vulnerable, keeping the many android users at risk. While the fix is there, deploying it on such a massive scale is becoming difficult. This has led to a fear that many people might have stopped trying to install patches.


Australian security expert, Robert Graham's research has shown that out of the 600,000 vulnerable servers identified by him post Heartbleed, 300,000 servers were still exposed, as recent as the end of June. While reason for this can easily be attributed to incompetency, but this will not be the truth.

One might think that all is well now, and a few broken servers are not going to affect anyone, but that is not how IT security works. There is threat of data loss, yes...but more than that, a server which is compromised is like a huge gap in your protective fence, which if not mended will sooner than later give entry to lurking cybercriminals. Therefore, securing IT infrastructure requires diligent vigilance. One must have the right security tools in place and perform continuous website security checks that will share regular security updates to the business owners.

Internet is a crucial part of our everyday life. It is vulnerable to cybercrimes and cybercriminals and the aftereffects of Heartbleed have increased their vulnerability. The onus to ensure the security of our IT infrastructure falls on us, and this is a part which is non-negotiable.

POODLE

Padding Oracle On
Downgraded Legacy
Encryption



Poodle



3.0 protocol
SSL

Google announced a vulnerability in the implementation of the SSL 3.0 protocol potentially compromising secure connections online. POODLE (Padding Oracle on Downgraded Legacy Encryption) was a new security hole in Secure Socket Layer (SSL) 3.0 that makes the 15-year-old protocol nearly impossible to use safely. The “Poodle” is a protocol flaw, not an implementation issue; every implementation of SSL 3.0 suffers from it. The TLS versions are not affected (neither is DTLS).

Poodle attack works in a chosen-plaintext context. The attacker is interested in data that gets protected with SSL, and he can:

1. Inject data of his own before and after the secret value that he wants to obtain;
2. Inspect, intercept and modify the resulting bytes on the wire.

The main and about only plausible scenario where such conditions are met is a Web context: the attacker runs a fake WiFi access point, and injects some Javascript of his own as part of a Web page (HTTP, not HTTPS) that the victim browses. The evil JavaScript makes the browser send requests to a HTTPS site (say, a bank Web site) for which the victim’s browser has a cookie. The attacker wants that cookie. The attack proceeds byte-by-byte. The attacker’s JavaScript arranges for the request to be such that the last cookie byte occurs at the end of an encryption block (one of the 8-byte blocks of 3DES) and such that the total request length implies a full-block padding.

Can you detect Poodle attacks on your network?

You don't! Since the most probable attack setup involves the attacker luring the victim on his network, not yours.

Although, on the server side, you may want to react on an inordinate amount of requests that fail on a decryption error. Not all server software will log events for such cases, but this should be within the possibilities of any decent IDS system.

Recommendations by Google:

The attack described above requires an SSL 3.0 connection to be established, so disabling the SSL 3.0 protocol in the client or in the server (or both) will completely avoid it. If either side supports only SSL 3.0, then all hope is gone, and a serious update required to avoid insecure encryption. If SSL 3.0 is neither disabled nor the only possible protocol version, then the attack is possible if the client uses a downgrade dance for interoperability.

Disabling SSL 3.0 entirely right away may not be practical if it is needed occasionally to work with legacy systems. Also, similar protocol version downgrades are still a concern with newer protocol versions (although not nearly as severe as with SSL 3.0).

The `TLS_FALLBACK_SCSV` mechanism from [draftietf-tls-downgradescsv00] addresses the broader issue across protocol versions, and we consider it crucial especially for systems that maintain SSL 3.0 compatibility.

The following recommendations summarize how `TLS_FALLBACK_SCSV` works:

TLS clients that use a downgrade dance to improve interoperability should include the value `0x56, 0x00` (`TLS_FALLBACK_SCSV`) in `ClientHello.cipher_suites` in any fallback handshakes. This value serves as a signal allowing updated servers to reject the connection in case of a downgrade attack. Clients should always fall back to the next lower version (if starting at TLS 1.2, try TLS 1.1 next, then TLS 1.0, then SSL 3.0)

because skipping a protocol version forgoes its better security. (With TLS_FALLBACK_SCSV, skipping a version also could entirely prevent a successful handshake if it happens to be the version that should be used with the server in question.) In TLS servers, whenever an incoming connection includes 0x56, 0x00 (TLS_FALLBACK_SCSV) in ClientHello.cipher_suites, compare ClientHello.client_version to the highest protocol version supported by the server. If the server supports a version higher than the one indicated by the client, reject the connection with a fatal alert (preferably, inappropriate fallback (86) from [draft-ietf-tls-downgrade-scsv-00]).

This use of TLS_FALLBACK_SCSV will ensure that SSL 3.0 is used only when a legacy implementation is involved: attackers can no longer force a protocol downgrade. (Attacks remain possible if both parties allow SSL 3.0 but one of them is not updated to support TLS_FALLBACK_SCSV, provided that the client implements a downgrade dance down to SSL 3.0.)

BASH



▄▄▄▄▄▄ BASH ▄▄▄▄▄▄

A major vulnerability in the Bourne Shell — called bash shell – has come to the fore. This vulnerability existed from a long time. Using the vulnerability, a client (or equivalently a server) can insert malicious commands into the server (or equivalent client) when the server (equivalently client) uses a bash script with inputs from environmental variables that are set by the client (or equivalently server).

The implications are huge as it affects all *NIX systems — Linux — Ubuntu, Debian, CentOS, others, UNIX with all its variants such as Solaris, BSD, NetBSD, others. Unix/Linux systems are so prevalent that they are used everywhere and the shell is one of the most common programs used in these systems.

To get into a bit more technical details, these are the details of the vulnerability. Some of this material is taken from <http://seclists.org/oss-sec/2014/q3/650>. Bash supports exporting not just shell variables, but also shell functions to other bash instances, via the process environment to (indirect) child processes. Current bash versions use an environment variable named by the function name, and a function definition starting with “(){}” in the variable value to propagate function definitions through the environment. The vulnerability occurs because bash does not stop after processing the function definition; it continues to parse and execute shell commands following the function definition. For example, an environment variable setting of

```
VAR={() { ignored; }; /bin/id
```

will execute /bin/id when the environment is imported into the bash process. (The process is in a slightly undefined state at this point. The PATH variable may not have been set up yet, and bash could crash after

executing `/bin/id`, but the damage has already happened at this point.)

The fact that an environment variable with an arbitrary name can be used as a carrier for a malicious function definition containing trailing commands makes this vulnerability particularly severe; it enables network-based exploitation.

Thus, for instance when you connect your mobile or a computer system to a Wi-Fi or a wired network, a DHCP client takes variables sent from a DHCP server while your system (mobile, desktop) connects to a network. These variables are used in a shell program. A rogue DHCP server can now play havoc with your system by running malicious commands on it.

A webserver such as apache/NGINX, when it runs CGI-scripts, takes input from the client via shell variables. Here, the client can insert malicious commands.

Right from routers, to all kinds of other systems, shell scripts are used, and this vulnerability can play havoc. In fact, the bash shell is so ubiquitous that it may be impossible to know the full extent of this vulnerability. You may be vulnerable even if you are using shell to connect to a remote system.

The way out is to upgrade your shell to the latest version. Patched bash shells are now out from various vendors.

The other workaround is to insert WAF signatures to block this vulnerability/exploit in case you are running a website. In case, you are running a program such as SSH over the network, an appropriate signature will have to be installed at layer 4 – in the IPS.

Indusface's core rule set has an exhaustive protection for "command injection" category of vulnerabilities, those core rules already protected users against most of the Bash centric vulnerabilities. We have added few more signatures for various customer environments to ensure highest level of customized security for existing IndusGuard WAF customers.

GHOST VULNERABILITY



On January 27, 2015, a serious weakness was found within the Linux operating system, which can potentially provide complete control over compromised system. Now given that Linux is still very popular with smartphones and servers, Indusface Research Team believes that it can be seriously threatening to businesses. Following is a brief guide on all the information you will need on the topic.

CVE-2015-0235 Basics

CVE-2015-0235 is being called the GHOST Vulnerability as it exploits glibc's GetHOST functions. It basically affects Linux glibc or GNU C library on versions prior to glibc-2.18. Now, GNU C Library is a core part of the Linux operating system in glibc 2.2 to glibc 2.17. With buffer overflow in glibc function `__nss_hostname_digits_dots()`, an attacker can exploit the bug even from a remote location with `gethostbyname*()` functions. Now that the DNS resolver and application are connected, it becomes easier to get IP address from a hostname. Many Linux distributions including, but not limited to the following may be affected.

- Debian 7
- CentOS 6 & 7
- Ubuntu 10.04 & 12.04
- Red Hat Enterprise Linux 6 & 7
- End of Life Linux Distributions

Risk Analysis

As the GHOST vulnerability can be exploited both locally and remotely, it becomes very easy to gain complete control over the compromised system. It has been found that an attacker can bypass almost every protection layer on both 32-bit and 64-bit systems, leaving server prone to all kind of brand and financial damage.

Affected Operating Systems

Our existing customers will get an alert through IndusGuard WEB scanning to monitor and defend their server assets. We have updated our scanning vectors to look for the GHOST vulnerability. Here's how others can look for glibc versions. For Ubuntu and Debian, check out the ldd version:

```
ldd --version
```

Look for the glibc version in the first line and match it with the following numbers. If yours is older than the following, patching is must.

- Debian 7 LTS: 2.13-38+deb7u7
- Ubuntu 10.04 LTS: 2.11.1-0ubuntu7.20
- Ubuntu 12.04 LTS: 2.15-0ubuntu10.10

For RHEL and CentOS too, look for ldd version.

```
ldd --version
```

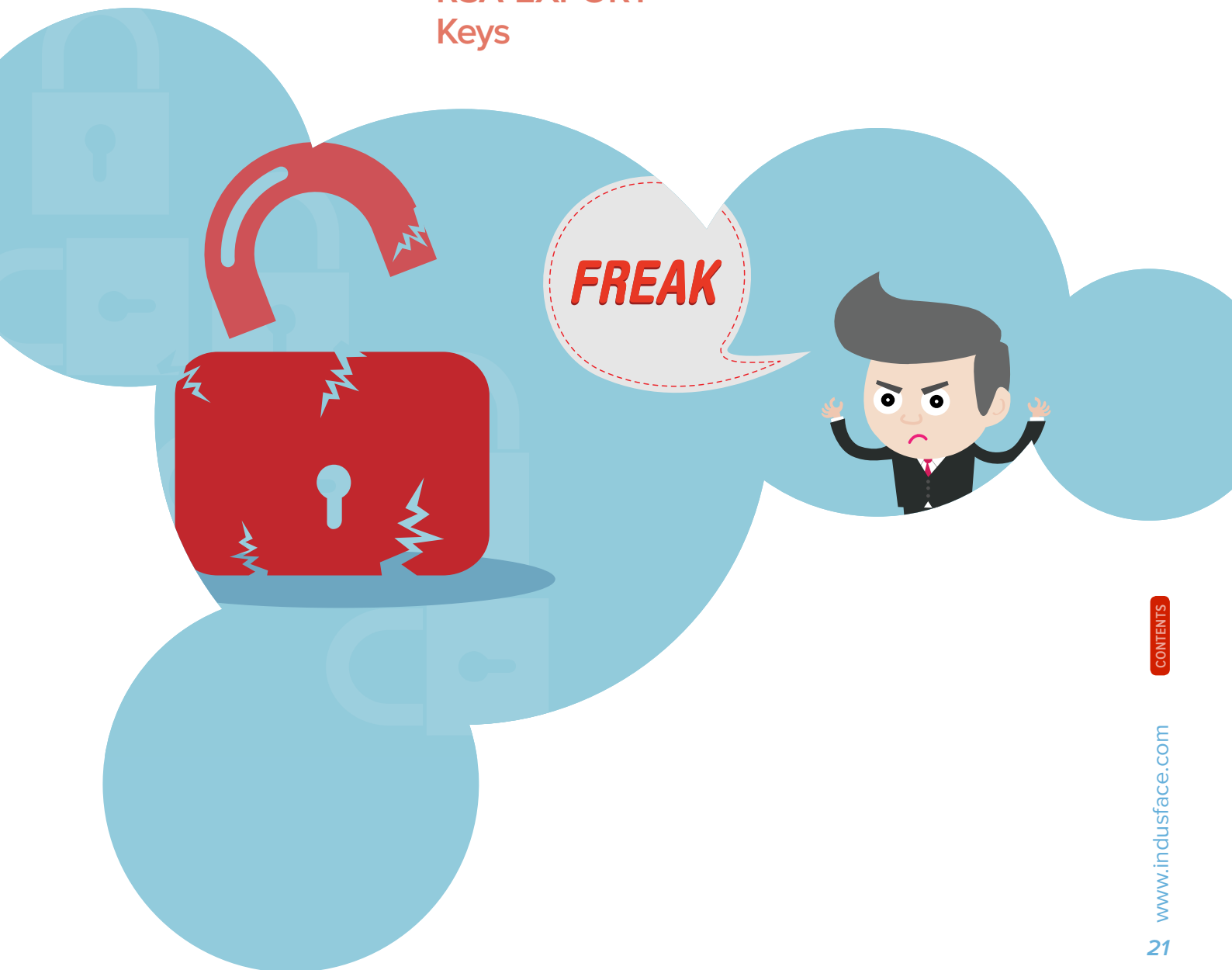
You should get the glibc from first line of the result. If it is more recent than 2.18, you do not need to worry. For older versions, patch is necessary.

Mitigation

Update glibc version using default package manager for OS. You can contact your license vendor and apply for a patch to get rid of the issue. Once the system has been updated, make sure that you check for the glibc version once again, just to be sure. Our research team is constantly reviewing the developments on the GHOST vulnerability and promises to come up with important details when required. You can also contact us to understand how IndusGuard WEB can help detect GHOST and several other vulnerabilities continuously.

FREAK

Factoring
Attack on
RSA-EXPORT
Keys



A security loophole from the early 90s, which nobody really remembers, has come back to haunt around 33% of all the websites and servers across the world in March 2015.

Termed as the “FREAK” vulnerability, CVE-2015-0204 stands for Factoring Attack on RSA-EXPORT Keys. It exposes many SSL clients including OpenSSL to weak encryption and theft of sensitive data within the communication channel. According to the French researchers who had reported this vulnerability, Android and Safari browsers are at severe risks of man-in-the-middle hacks. Meanwhile, Microsoft has also confirmed that it affects all currently supported Windows versions too.

Where did the FREAK come from?

Somewhere in the early 90s, the US government restricted native companies from exporting any machines that utilized strong encryption for security reasons. They believed that other countries could have used the encryption against them.

At the same time, there were no restrictions on easy-to-break or weak encryption products, which were exported in huge numbers. It is believed that the National Security Agency also wanted to decrypt foreign encryption at that time. Clearly it was not an easy task to break the encryption as it required supercomputers and access to few other resources that the US government had at their disposal.

Towards the end of the 90s, the US government lifted those export restrictions and almost everyone forgot about those weaker encryption ciphers. Somehow (cryptographers are still researching on whys and hows of it) those low-grade encryption modes are still found in many products that use unpatched OpenSSL, especially in Android and Apple devices.

How is the FREAK vulnerability exploited?

French cryptographic team, after discovering the vulnerability, devised a plan and was able to trick the browsers into accepting weak encryption modes for hacking into several website. It is being assumed that more than 5 million websites with SSL encryption pad lock are vulnerable to such exploitation today, which also includes the supposedly 'secured' sites and cloud providers like those of FBI, NSA, IBM and, Symantec.

FREAK or CVE-2015-0204 is basically a Factoring Attack on RSA-EXPORT Keys with comparatively weak encryption. While it's true that back in the 90s only a few agencies had access to supercomputers and other resources, today the scenario has changed with cloud computing services like Amazon's EC2 bringing advanced computing for hire within everyone's reach. Here's how an attacker can exploit FREAK using that.

- A man-in-the-middle attacker forces connection to use weaker RSA cyber suite with an altered message.
- The server responds with a 512-bit export RSA key, signed with its long-term key.
- Victim accepts outdated key due to the vulnerability.
- Attacker factors RSA modulus for decryption key.
- When victim encrypts a file, attacker decrypts it and accesses the information in plain text.

Quite clearly such an exploitation shatters HTTPS security and open gates to private key, login cookies, and passwords. Not only can the attacker access all the sensitive information, but FREAK also allows him to inject any command.

How can you ensure security?

While a lot of security analysts are arguing about the severity of FREAK, no one can argue that any vulnerability that threatens your sensitive information and poses command injection risks has to be taken seriously.

According to Apple's spokesman Ryan James, they have developed a software update to remediate the vulnerability, which would be pushed out next week. And Google spokeswoman Liz Markman said the patch has been provided to partners, but she did not comment on its availability.

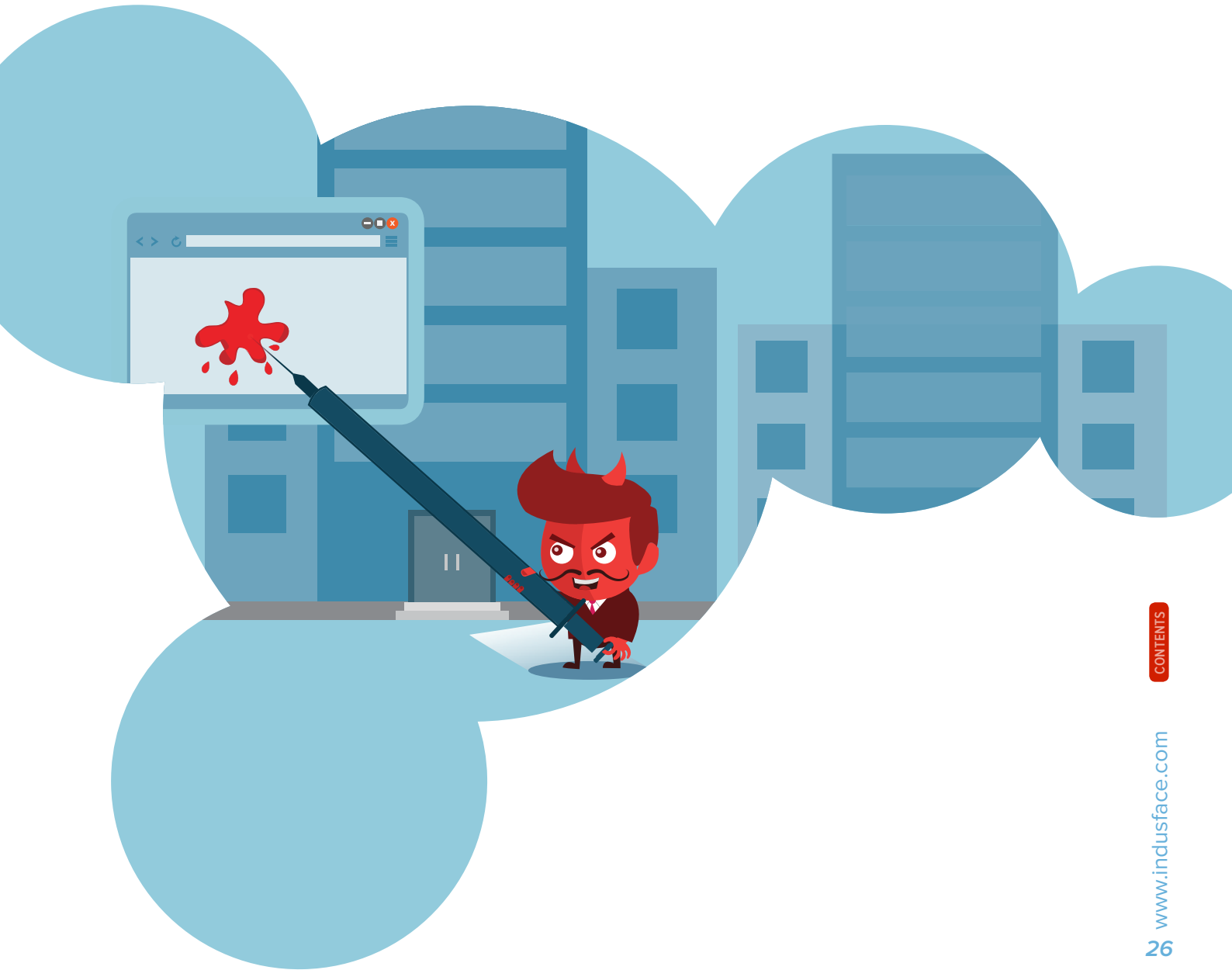
Meanwhile, individual users are advised to prefer Google Chrome and Firefox for their stronger encryption technology over other browsers.

As for the organizations, it is extremely important to keep the applications and servers protected from man-in-the-middle exploitations. A vulnerability scanner is recommended to point out weakly encrypted handshakes with the client that could lead to injection. We have already updated the IndusGuard Web scanners and WAF to detect, report and protect from the FREAK vulnerability.

An illustration on a light blue background. A hand holds a smartphone in the center. The screen of the phone displays the text 'AND THE USUAL SUSPECTS THAT MAKE IT TO HEADLINES' in a bold, blue, sans-serif font. Surrounding the phone are several light blue, fluffy clouds. From each cloud, a pair of stylized, black-rimmed eyes with white pupils and black pupils is looking out. The eyes are positioned at the top left, bottom left, and right sides of the phone. The hand holding the phone is a simple, orange-brown color with a white circle on the back of the hand.

**AND THE
USUAL
SUSPECTS
THAT MAKE
IT TO
HEADLINES**

SQL INJECTION



SQL injection is a code injection technique, used to attack data driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker).

SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

Normal Scenario: `username = author and password = author123`

SQL Query:

```
SELECT * FROM AUTH_TABLE WHERE user ='author' and passw = 'author123'
```

Attacking Scenario: `username = author and password = ' or '1 ='1`

SQL Query:

```
SELECT * FROM AUTH_TABLE WHERE user ='author' and passw = ' or '1 ='1'
```

The above line says, password equals to null or 1=1 (which is universal truth), so the attacker modifies logical AND to logical OR.

The best way to find out if an application is vulnerable to injection is to verify that all use of interpreters clearly separates untrusted data from the command or query. For SQL calls, this means using bind variables in all prepared statements and stored procedures, and avoiding dynamic queries.

Checking the code is a fast and accurate way to see if the application uses interpreters safely. Code analysis tools can help a security analyst find the use of interpreters and trace the data flow through the application. Manual penetration testers can confirm these issues by crafting exploits that confirm the vulnerability.

Automated dynamic scanning which exercises the application may provide insight into whether some exploitable injection problems exist. Scanners cannot always reach interpreters and can have difficulty detecting whether an attack was successful.

How to prevent injections like SQL:

Preventing injection requires keeping untrusted data separate from commands and queries.

1. The preferred option is to use a safe API which avoids the use of the interpreter entirely or provides a parameterized interface. Beware of APIs, such as stored procedures, which appear parameterized, but may still allow injection under the hood.
2. If a parameterized API is not available, you should carefully escape special characters using the specific escape syntax for that interpreter. OWASP's ESAPI has some of these escaping routines.
3. Positive or "whitelist" input validation with appropriate canonicalization also helps protect against injection, but is not a complete defense as many applications require special characters in their input. OWASP's ESAPI has an extensible library of white list input validation routines.

Here are some recommended best practices to prevent injection exploits:

1. Make white box testing and secure code review an integral part of the development cycle.
2. Manual penetration testing should be part of the QA cycle.
3. Automated along with periodic manual checks should be part of the operations process after deployment to keep up with new vulnerabilities and the automated scanning can help perform basic security sanity checks. It is very common to have injection issues introduced even with minor code changes as such changes may not be subjected to a full-fledged security review process in the development cycle.

4. In situations where fixing the code of detected vulnerabilities cannot be done in a timely manner a WAF can be used for virtual patching after deployment.

Last Year: Wall Street Journal Got SQL-ed

A hacker has claimed to infiltrate the Wall Street Journals' website by exploiting a vulnerability which existed in a web based graphics system. The newspaper acknowledged this news on 22nd July's evening. Apparently the hacker had also laid claim on the breach of 'WSJ' and 'Vice', both. The hacker in question here, goes by the name of 'w0rm' and has offered to sell both databases full of user information and the credentials necessary to control the WSJ's purportedly breached server.

To substantiate his claim of the dual breach, the hacker had posted screenshots of both WSJ and Vice on Twitter. The photos are no longer available.

The newspaper has said that the system was breached offline and the breach did not affect the customers or customer's data.

According to a security researcher, who analyzed w0rm's screenshot, the hacker was able to gain entry into the graphics system network via an SQL injection vulnerability. This could have given w0rm access to 23 other databases which existed on the same server, but to confirm and say that w0rm has that access, is hard.

w0rm had offered to sell the stolen data for one bitcoin (\$600 /£365).

This has been second attack on WSJ in a week. On Sunday, 20th July, hackers had gained access to their Facebook page, soon after unfortunate Malaysia Airlines plane crash. The hackers posted fake news alerts about the US President's plane possibly crashing over Russian airspace.

More on w0rm, the hacker who loves big data!

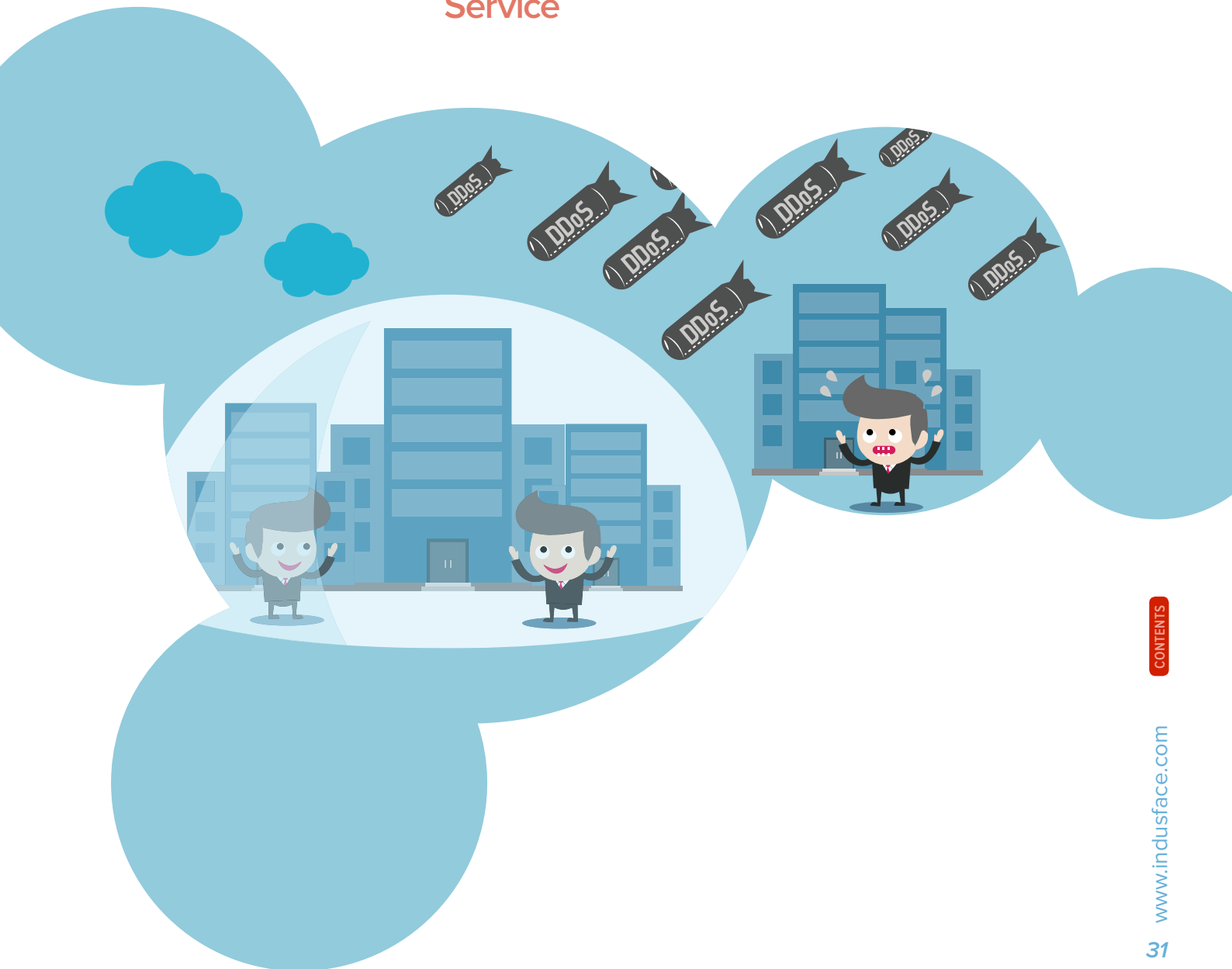
The hacker was previously known as rev0lver and has been described as highly motivated towards hacking big companies and brands with big databases and users, and then sell them further to other hackers. The data is sold on an online marketplace called 'w0rm.in', run by the perpetrator itself.

In a similar news, on 12th July, w0rm tweeted a screenshot, which apparently showed contents of the CNET database. CNET soon after confirmed that they had experience a cyber-attack and multiple servers had been breached. One million emails, usernames and encrypted passwords were stolen by the hacker, and following the same pattern as in WSJ hack, w0rm tweeted the availability of the database for one Bitcoin.

CNET later reported that w0rm had been identified as being a representative for a group of Russian hackers. They also said that the hacker no intention of decrypting the passwords or complete the mentioned sale of the database, and the only reason why w0rm had offered to sell the database for one bitcoin, was to gain attention. In December 2013, w0rm, which at that time was known as rev0olver, had tried to sell FTP credentials for BBC's server.

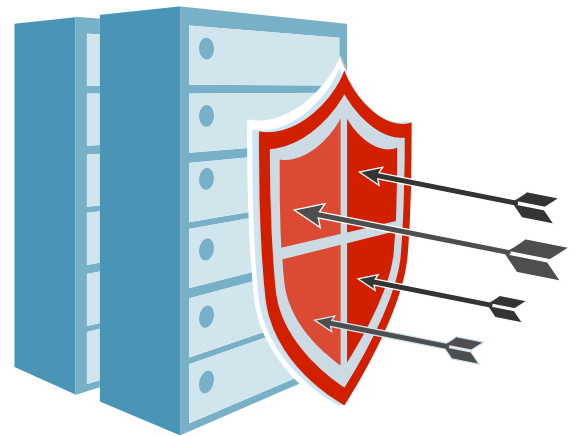
DDoS

Distributed
Denial of
Service



A distributed denial of service (DDoS) attack is one in which a multitude of compromised computers attack a single target, thereby stalling traffic for the legitimate users of the targeted system. The large flow of requests from the compromised systems, to the targeted system essentially forces the target system to shut down or report as out of service due to bandwidth issues, thereby paralyzing the targeted system.

DDoS attacks have rapidly become hacker's choice of attack, with evidently many major businesses falling at the receiving end. On June 10th, Evernote, popular note taking and web clipping saving service, became a victim of a similar attack. As a result, members were unable to synchronize their filings. The very next day, Feedly, a very popular news aggregator which provides content from various online sources at one place, was attacked. It was again, a DDoS attack, which caused the service to be unavailable for hours together. This attacks involved demand for ransom from the attackers to which Feedly refused. At 3:07 PT, Feedly announced that the attack had been neutralized, but within hours of this, the site reported being under the fire again. They were targeted by a second DDoS attack, which again caused their site to go down.



Last Year: DDoS Attacks Hit the World Cup!

While football fever struck worldwide, a major DDoS attack struck the official government World Cup website, which went down for more than a day. The latest name in this list of DDoS victims was of Hong Kong Democracy Poll, where attack was fended off by diverting most of the traffic to sinkholes. But the problem with sinkholing or black-holing is that though it diverts the traffic to a sinkhole where it is discarded, segregation between good and bad traffic cannot be done. This means that all traffic, whether good or bad, is discarded. While DDoS is bad news for organizations, resorting to sinkholing cannot be considered as an alternative.

XSS

Cross
Site
Scripting



Cross Site Scripting (XSS) is an attack in which an attacker exploits vulnerability in application code and runs his own JavaScript code on victim's browser. The impact of an XSS attack is only limited to the potency of the attacker's JavaScript code.



Stored XSS - Stored XSS are the ones where the injected code is permanently stored on the target servers, such as in a database, message forum, visitor log, comment field, etc. The victim retrieves the malicious script from the server when it requests the stored information.

Reflected XSS - Reflected XSS are the ones where the injected code is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request. Reflected XSS are delivered to victims via another route, such as in an e-mail message, or on some other web server. When a user is tricked into clicking on a malicious link or submitting a specially crafted form, the injected code travels to the vulnerable web server, which reflects the attack back to the user's browser. The browser then executes the code because it came from a "trusted" server.

DOM-based XSS - DOM Based XSS is an XSS attack wherein the attack payload is executed as a result of modifying the DOM "environment" in the victim's browser used by the original client side script, so that the client side code runs in an "unexpected" manner.

That is, the page itself (the HTTP response that is) does not change, but the client side code contained in the page executes differently due to the malicious modifications that have occurred in the DOM environment.

Last Year: **XSS Haunted Amazon and WordPress**

In early September, last year, security consultant Benjamin Mussler had warned that Kindle e-book library was vulnerable to a Cross-Site Scripting vulnerability. It came to light that Amazon was aware of this vulnerability and had fixed it earlier, but in July, Amazon introduced a new version of the “Manage Your Kindle” web application. It seems, that they re-introduced the vulnerability in this version, which surprisingly should have been fixed long ago. The XSS flaw allowed malicious users to inject victim’s account through e-book meta data with malicious code, which would be executed as soon as the victim accessed Kindle Library Web page. The hackers could then access and steal victim’s Amazon account cookies.

This came as a huge reputational blow to a company like Amazon, with experts raising concern on the fact that they ignored addressing this problem way earlier. Even though this problem affected only the user using free versions of e-books from random torrents or malicious websites, Amazon had to face a lot of heat for coming out as a vendor unable to keep its customer’s personal information secure.

WordPress was another recent victim to an XSS flaw, which they patched in November last year. The XSS vulnerability in the comment boxes of WordPress posts and pages, could be exploited by an attacker to create comments with malicious JavaScript code embedded in them. These would then get executed by the browsers of users seeing these comments.

“In the most obvious scenario the attacker leaves a comment containing the JavaScript and some links in order to put the comment in the moderation queue,” said Jouko Pynnonen, the security researcher



who found the flaw, in an advisory. “When a blog administrator goes to the Dashboard/Comments section to review new comments, the JavaScript gets executed. The script can then perform operations with administrator privileges.”

The 3.9.3, 3.8.5 and 3.7.5 updates from WordPress addressed this XSS vulnerability.

How to protect yourself from XSS attacks

There are many ways in which a web application can be protected from XSS attacks:

1. Proper validation of headers, query strings, cookies, hidden fields and all other parameters, to decide what to be allowed. Positive input validation is recommended here as opposed to negative input validation. I.e it should be checked that what inputs are allowed, rather than focusing on what inputs are not allowed.
2. In the event of displaying user supplied data, the data should be encoded.
3. You can use tools to test a code for XSS vulnerability, before deploying it on live site.
4. Developers should refer to security control libraries like OWASP's Enterprise Security API.
5. WAFs help in detecting and blocking attacks against XSS attacks. They block keywords that lead to XSS, therefore a hacker trying to XSS a site inputs JavaScript or other similar scripts is stopped by blocking these scripts.



TOTAL APPLICATION SECURITY

that **DETECTS,**
PROTECTS and
MONITORS

How Secure is your
Website?

Get a free
detailed analysis
done today.

[I WANT A FREE TRIAL](#)

"Indusface is an innovative, fast growing information security company, trusted by fortune 500 organizations across the globe and catering to more than 700 customers worldwide across different verticals. Indusface helps safeguard web and mobile applications using its flagship product IndusGuard, giving customers the distinctive edge of having total application security. Indusface has been recognized by Gartner in 2013 and positioned on the Magic Quadrant for Application Security Testing. Indusface has also been recognized by NASSCOM DSCI, Deloitte Technology, Red Herring Top 100 Asia and ChannelWorld 100. It has been empanelled by prestigious independent bodies, PCI ASV and CERT-IN."



Gartner ■ Magic Quadrant
for Application Security Testing
- Niche Player

Deloitte.
Tech Fast 50
India & 500 Asia



FINALIST -
DSCI Excellence
Awards 2013

