acunetix

Web Application Vulnerability Report

2016

Introduction

Welcome to the 2016 edition of the Acunetix Web Application Vulnerability Report.

This document presents the second *Web Application Vulnerability Report*, an annual effort from the Acunetix Team. In this report, Acunetix will present data gathered, aggregated and analyzed throughout the period of 1st April 2015 to 31st March 2016 to illustrate the state of security of web applications and network perimeters.

By analyzing scan results on Acunetix' Online Vulnerability Scanner (OVS) platform, we are able to identify current and emerging patterns in the web application security space. With over 61,000 web and network security scans run over a two-year period, Acunetix is uniquely positioned to observe such trends.

Web app vulnerabilities have rapidly increased in the past 12 months as companies demand faster web application release cycles to satisfy staff and customers. Web application vulnerabilities are dangerous for organizations as they risk not only brand and reputational damage, but data breaches and the major fines associated with these. The findings continue to reaffirm the widely held understanding that the web application vector is a major, viable and low-barrier-to-entry vector for attackers; be they financially motivated, "hacktivists", nation-state attacks or threat actors.

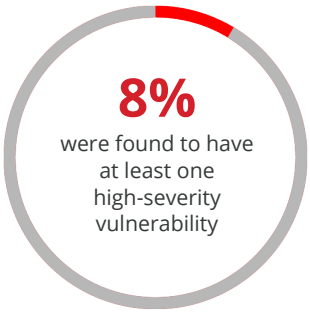
The threat landscape is changing—the web stack, has evolved to serve-up rich experiences directly within the browser. As a result of the versatility and platform agnosticism that web applications provide, web applications and web services are ever increasingly replacing legacy applications, and as a consequence, widening attackers' exploitation opportunities; especially since traditional network-layer-only security controls such as firewalls and signature-based intrusion prevention and detection systems (IPS/IDS) have little, or no role to play in detecting and stopping an attack occurring via the web application vector.

This report aims to shed a light on the state of web and perimeter network security based on the analysis undergone. While this research found a minor, but encouraging reduction in security vulnerabilities such as SQL injection and Cross-site Scripting, web application vulnerabilities still reign supreme and are worryingly on the rise. Now the majority of web apps (55%) contain a high severity vulnerability, up from 46% last year.

The Acunetix Team

Web Applications

Perimeter Network Assets



Methodology

The data aggregated and analysed in this report was gathered from automated web and network perimeter scans run on the Acunetix Online Vulnerability Scanner platform, over the period of one year, starting 1st April 2015 to 31st March 2016. Evaluation scans on the intentionally vulnerable Acunetix test websites were omitted for the scope of this analysis.

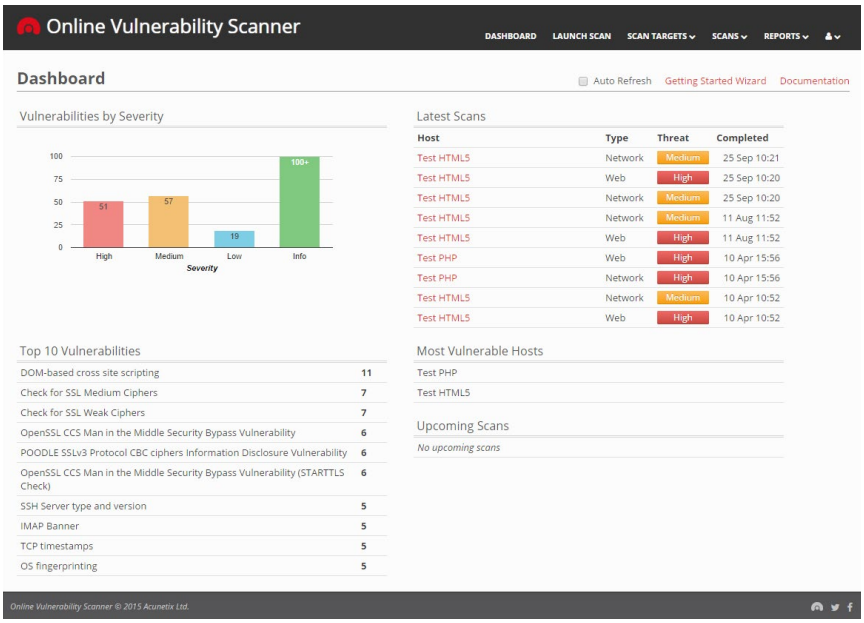
For the purpose of this analysis, a random sample of 5,700 subscribers who have successfully scanned one or more Scan Targets were randomly selected out of a possible 37,500 subscribers.

How an Automated Web Scan Works

The scanning process comprises of three stages—Crawling, Scanning and Reporting.

Crawling

During the crawling stage, Acunetix Vulnerability Scanner analyzes the structure of the web application being scanned by leveraging its DeepScan crawling and scanning engine.



DeepScan not only looks for links and inputs, but also executes JavaScript and can interact with HTML5-based web applications just like a user in a modern web browser would. This means that modern client-side applications leveraging JavaScript frameworks like Angular JS, React, Ember.js can be properly tested.

Scanning

The scanning stage is where Acunetix Vulnerability Scanner tests the web application for over 3000 vulnerabilities, some relating specifically to web server security, misconfigurations, and information disclosure; while the, large majority focus on testing inputs on a page for vulnerabilities. The scanner can automatically test JSON, XML and Google Web Toolkit (GWT) input vectors in addition to the typical GET and POST parameters.

Reporting

The third final stage of a scan is reporting, where, after a scan is complete, vulnerability alerts are reported, complete with detailed information about vulnerabilities in question, remediation advice and links to other online references.

The Dataset

The data analysed in this report is gathered from automated web and network perimeter scans run on the Acunetix Online Vulnerability Scanner platform.

This dataset focuses predominantly on high and medium-severity vulnerabilities found in web applications as well as perimeter network vulnerability data.

17, 962
Network Scans

27, 248
Web Scans

5, 718
Scan Targets

Average/Month

347, 000
files scanned

204,000
directories scanned

232, 000, 000
HTTP requests done

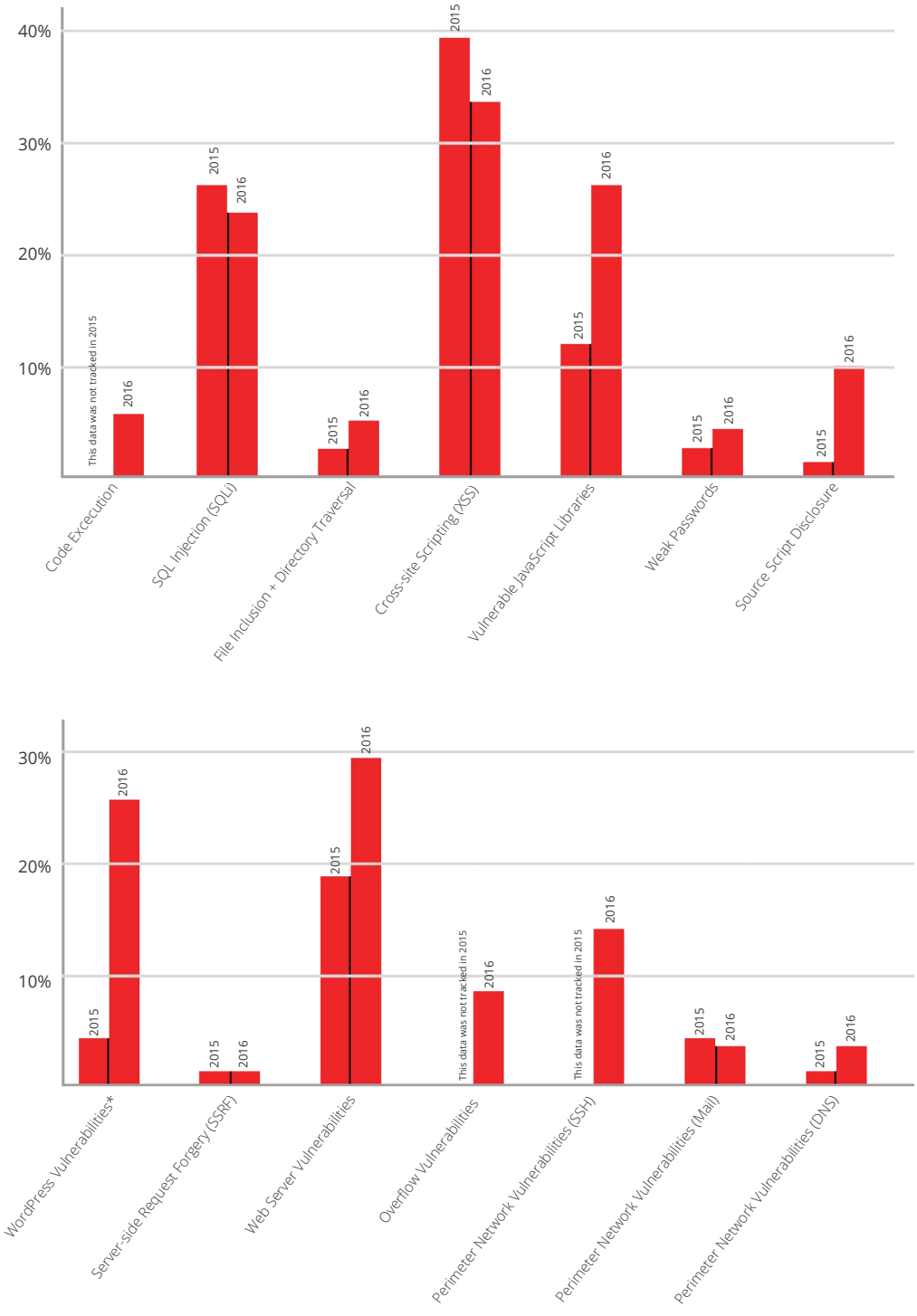
208, 000
total alerts discovered

Vulnerabilities at a Glance

What Changed and What Hasn't

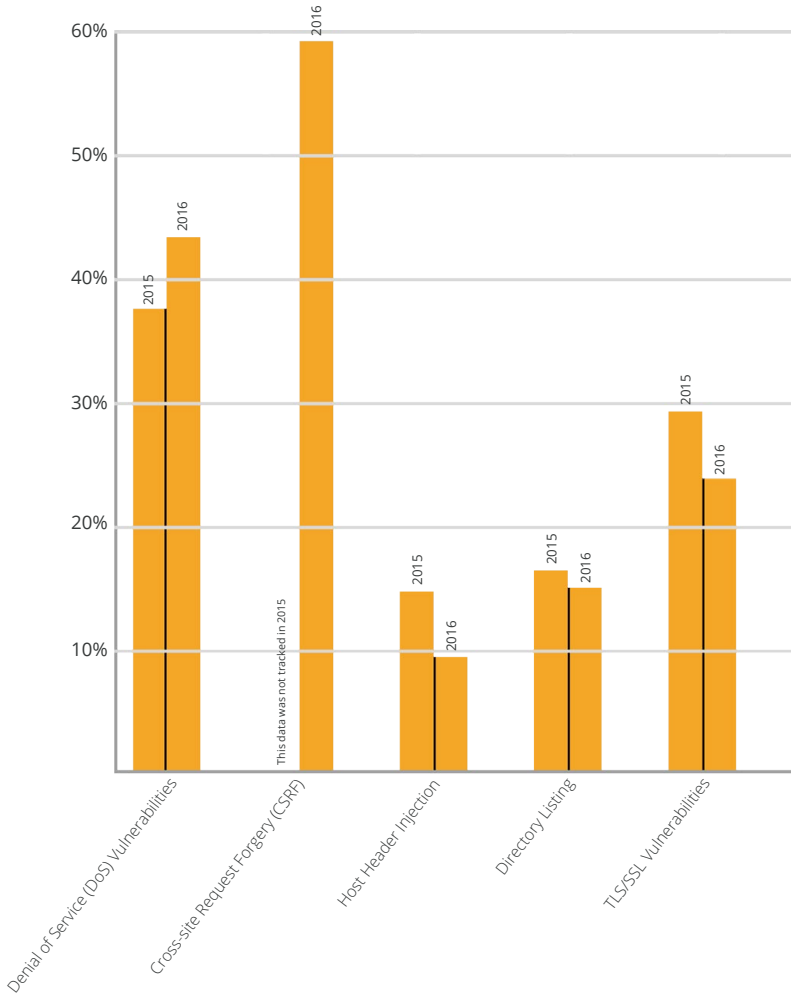
By comparing this dataset with results obtained last year, we can observe areas of improvement and regression in the amount of vulnerabilities by class.

Vulnerabilities by Type - High Severity

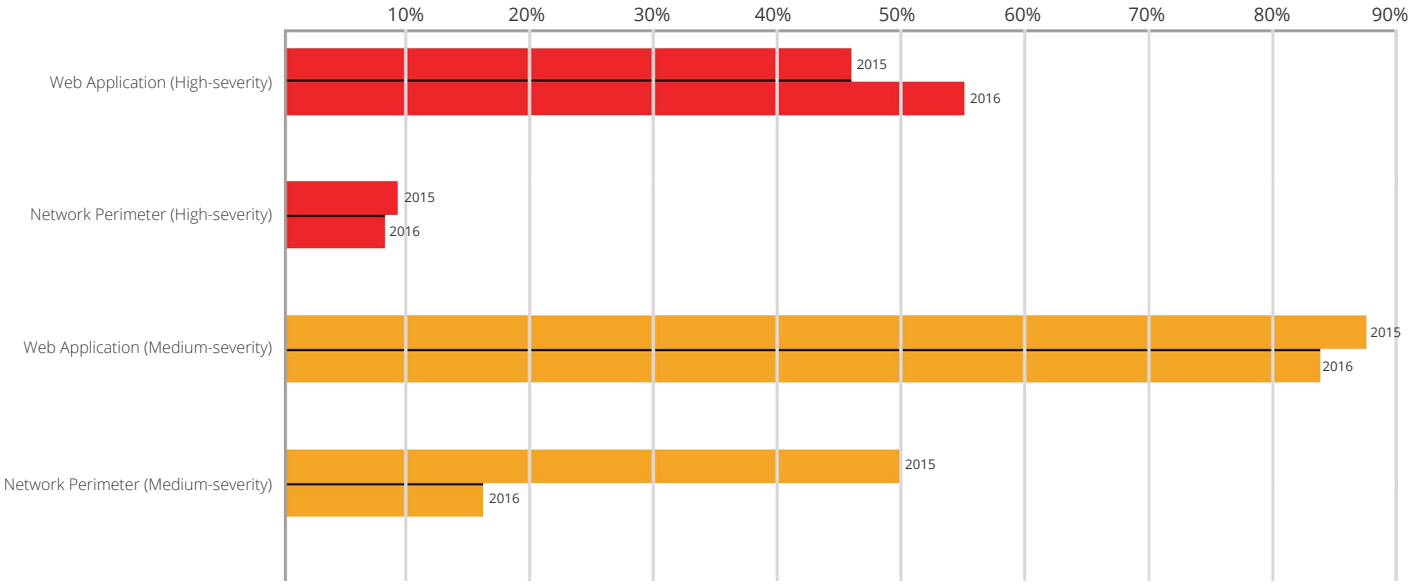


*N.B. The increase in WordPress Vulnerabilities in this case, can be attributed to the fact that the latest version of Acunetix (v10.5) used in the purpose of this analysis, includes many more WordPress vulnerability checks than previous Acunetix version 9 had used in 2015.

Vulnerabilities by Type - Medium Severity



Vulnerabilities by Paradigm and Severity



Vulnerability Severity

Severity is a metric for classifying the level of risk which a security vulnerability poses.

The severity level of a vulnerability is assigned based on the security risk posed to an organization should the vulnerability be exploited, as well as the degree of difficulty involved in exploiting it. The result of a successful attack by exploiting a vulnerability could vary from denial of service and information disclosure, to a complete compromise of applications or systems.

The following provides a description of what the results in this analysis consider to be the impact of each vulnerability severity level.

High-severity	Medium-severity	Low-severity
An attacker can **fully** compromise the confidentiality, integrity or availability, of a target system without specialized access, user interaction or circumstances that are beyond the attacker's control. Very likely to allow lateral movement and escalation of attack to other systems on the internal network of the vulnerable application.	An attacker can **partially** compromise the confidentiality, integrity or availability, of a target system. Specialized access, user interaction, or circumstances that are beyond the attacker's control *may be required* for an attack to succeed. Very likely to be used in conjunction with other vulnerabilities to escalate an attack.	An attacker can **limitedly** compromise the confidentiality, integrity or availability, of a target system. Specialized access, user interaction, or circumstances that are beyond the attacker's control *is required* for an attack to succeed. Needs to be used in conjunction with other vulnerabilities to escalate an attack.

Results

Code Execution	Severity	High

Description

Remote Code Execution (RCE) is a very dangerous vulnerability that allows an attacker to execute arbitrary commands on the target web server (usually in a target process). The ability to trigger arbitrary code execution from one machine on another, especially over the Internet, is often referred to as remote code execution (RCE).

Impact

A code execution bug is arguably the most severe effect a vulnerability can cause since it potentially allows an attacker to take over the system entirely, from where an attacker can likely achieve *lateral movement*, taking note of resources on the network and seeking opportunities for collecting additional credentials or *privilege escalation*.

5.67% of targets sampled were found to be vulnerable to code execution. This is a very troubling figure, given the severity of the vulnerability. It is strongly recommended to refrain from using user input to execute any commands within an application, however, if you must do so, user input needs to be properly validated and escaped to prevent code execution.

SQL Injection	Severity	High

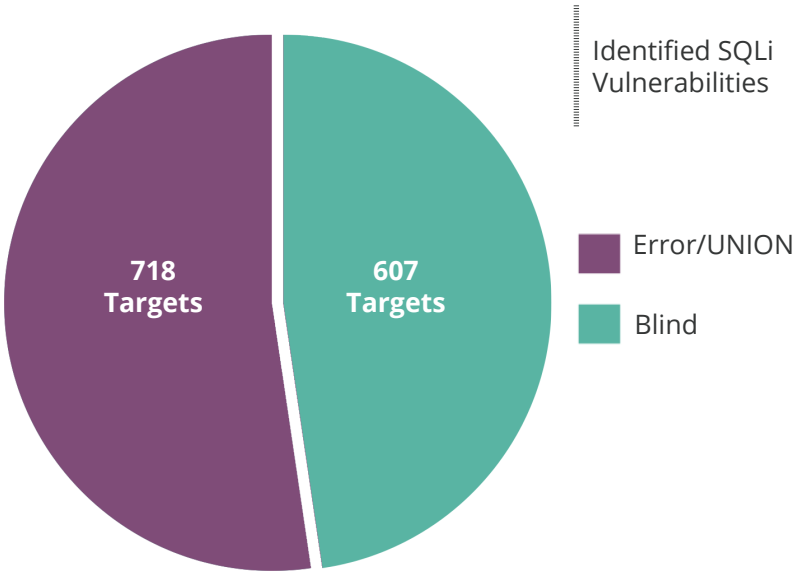
Description

SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements (also commonly referred to as a malicious payload) that control a web application's database server (also commonly referred to as a Relational Database Management System – RDBMS).

Since an SQL injection vulnerability could possibly affect any website or web application that makes use of an SQL-based database, the vulnerability is one of the oldest, most prevalent and most dangerous of web application vulnerabilities.

An attacker taking advantage of an SQLi vulnerability is essentially exploiting a weakness introduced into the application through poor web application development practices. This allows attackers to send SQL commands to the web application, allowing them to gain unauthorized access to data held in the backend database.

By leveraging an SQL injection vulnerability, given the right circumstances, an attacker can use it to bypass a web application's authentication and authorization mechanisms and retrieve the contents of an entire database. SQL injection can also be used to add, modify and delete records in a database, affecting data integrity.



To such an extent, SQL injection can provide an attacker with unauthorized access to sensitive data including, customer data, personally identifiable information (PII), trade secrets, intellectual property and other sensitive information.

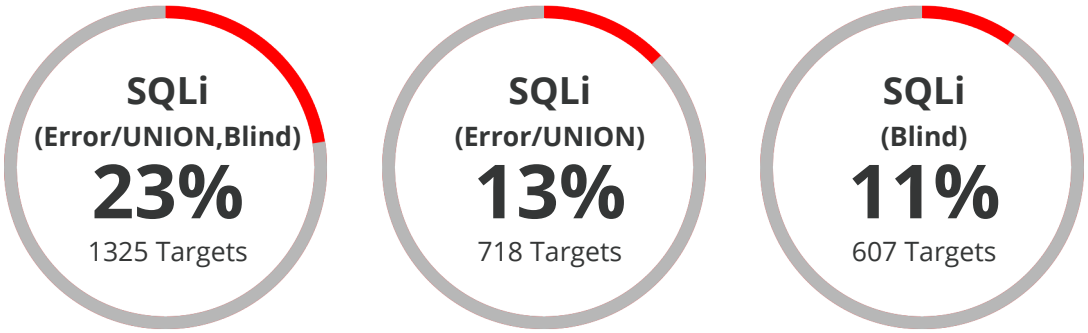
Blind SQL Injection is a kind of SQLi attack that is used when the results of an injection attack is not visible to the attacker. This does **not** imply that SQL injection is not possible, however, an attacker will need to find some other way of extracting data out of the database.

While a Blind SQLi attack does not display data within the response from the server, the attacker is able to retrieve data from the database by analyzing the results of a logical statement injected into the SQL query, for instance by asking the database to 'wait' a specified amount of time if a condition is true.

Impact

While SQLi is mostly used to steal data from the database, the vulnerability can be escalated further, especially if the permissions on the database are not correctly configured. For example, the attacker can inject a query that causes some tables to be deleted from the database, effectively causing a DoS attack.

An attacker can also potentially deploy a web shell onto the server and subsequently take over the server, and even pivot into other systems as a result of SQLi.



23% of sampled targets were vulnerable to at least one SQL injection vulnerability. The severity and ease of exploitation, combined with the maturity of exploitation tools targeting SQL injection makes this figure worrying; especially when considering how well understood and documented this vulnerability is.

▼ 3% However, all is not bleak with regards to SQL injection, this analysis has registered a 3% drop from last year, which indicates that things are very slowly moving in the right direction, however, as is the case with most other vulnerabilities in this report, SQL injection is clearly not a thing of the past and a lot more still needs to be done to address it.

File Inclusion and Directory Traversal	Severity	High

Description

File inclusion and directory traversal vulnerabilities could allow an attacker to access restricted files and directories outside of a web server's root directory. In the case of file inclusion vulnerabilities, the vulnerable application would not just allow the file to be read, but it would also execute its contents, while directory traversal only allows the reading of files.

Impact

File inclusion and directory traversal vulnerabilities are very dangerous since they both allow disclosure of sensitive files, including source code, secrets and sensitive configuration values. In the case of file inclusion vulnerabilities, this is also extended to the execution of interpreted code (such as PHP), and therefore, if combined with a file upload or arbitrary file write vulnerability (possibly even through SQL injection), file inclusion vulnerabilities could be escalated to code execution by an attacker using what is known as a web shell.

2% of sampled targets were found to be vulnerable to file inclusion

3% were found to be vulnerable to directory traversal.

▲ 1% These figures are on the rise from last year's 1% figure (both for file inclusion and directory traversal)—which is of some concern, especially for file inclusion, through which an attacker could potentially execute code given the right conditions. Both file inclusion and directory traversal vulnerabilities, like most other web vulnerabilities arise from the implicit trust web developers place in user input.

Cross-site Scripting	Severity	High

Description

Cross-site Scripting (XSS) is a vulnerability wherein client-side code injection occurs, predominantly through the use of JavaScript due to its prevalence in most browsing experiences.

Cross-site Scripting can be classified into four major categories: **Stored XSS, Reflected XSS, DOM-based XSS** and **Blind XSS**.
In all cases with XSS, the goal of an attacker is to get a victim to inadvertently execute a maliciously injected script. The malicious script is often referred to as a malicious payload, or simply a payload.

Stored (Persistent) XSS attacks involve an attacker injecting a script (referred to as the payload) that is permanently stored (persisted) on the target application (for instance within a database, in a comment field or in a forum post).

Reflected XSS attacks involve an attacker luring a victim to inadvertently make an HTTP request containing an XSS payload to a web server, usually achieved through phishing or other social engineering attacks. Once sent to the web server, the payload is then reflected back in such a way that the HTTP response includes the payload from the HTTP request.

DOM-based XSS is an advanced type of XSS wherein a payload is executed as a result of legitimate client-side JavaScript modifying the Document Object Model (DOM) in a victim's browser. In contrast to the other types of XSS, with DOM-based XSS, the HTTP response itself does not typically change, but rather client side code designed to process elements in the DOM, executes the malicious payload that has been injected in the DOM elements processed by the vulnerable JavaScript code.

When a web application is vulnerable to XSS, it will load the attacker-supplied content from a source that the application implicitly trusts, without properly encoding it. With stored and blind XSS, implicitly-trusted data is loaded from a datastore (such as a database or cache); with reflected XSS, the implicitly-trusted data is loaded from the HTTP request; and with a DOM-based XSS, implicitly-trusted data is loaded from a DOM-XSS source within the browser's DOM.

In every case, XSS would result in the browser interpreting the attacker's payload as legitimate JavaScript code, and subsequently executing it. It is important to note that an XSS vulnerability can only exist if the attacker's payload ultimately gets rendered in the victim's browser.

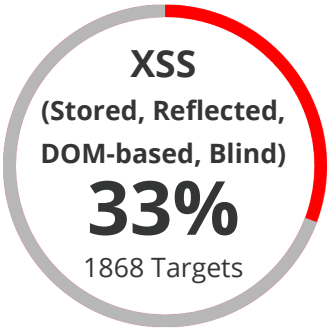
Impact

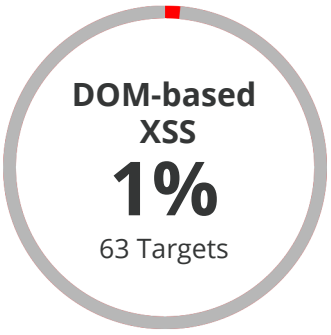
The consequences of an XSS attack may not be immediately obvious, especially since modern web browsers run JavaScript in a tightly controlled environment and since JavaScript has limited access to the user's operating system and the user's files.

However, when considering that malicious JavaScript has access to all the same objects as the rest of the web page, including access to cookies which are often used to store session tokens, if an attacker can obtain a user's session cookie, they can then impersonate that user.

Furthermore, JavaScript can read and make arbitrary modifications to the browser's DOM (within the page in which that script is running).

JavaScript can also be leveraged to send HTTP requests with arbitrary content to arbitrary destinations, and in modern browsers, can leverage HTML5 APIs such as accessing a user's geolocation, webcam, microphone and even the specific files from the victim's file system. While such APIs require the victim's opt-in, XSS in conjunction with some clever social engineering can bring an attacker a long way.





33% of sampled targets were vulnerable to at least one Cross-site Scripting vulnerability. The combination of XSS and social engineering, allow attackers to pull off advanced attacks including cookie theft, keylogging, phishing and identity theft. Critically, XSS vulnerabilities provide the perfect ground for attackers to escalate attacks to more serious ones.

▼ 6% Cross-site Scripting vulnerabilities have seen a 6% drop from last year, which is a sign of improvement, however, clearly, XSS is still a major issue plaguing web security. As JavaScript becomes ever more powerful, XSS becomes increasingly more dangerous.

Vulnerable JavaScript Libraries

Severity: High

Description

JavaScript has become a ubiquitous and every-day part of the web. Therefore, in order to make development faster and easier, many web applications rely on JavaScript libraries to avoid 'reinventing the wheel'. Unfortunately, many of these JavaScript libraries contain vulnerabilities, and therefore need to be updated to their latest version.

Impact

Running vulnerable JavaScript libraries exposes web applications to security vulnerabilities, most commonly being Cross-site Scripting vulnerabilities. Using components and libraries with known vulnerabilities can pose a significant risk to a web application and JavaScript libraries are certainly no exception.

27% of sampled targets were found to be making use of vulnerable JavaScript libraries within their web applications. Vulnerable JavaScript libraries open up the web application in concern to Cross-site Scripting (XSS) attacks. By far the most frequently encountered vulnerable JavaScript library was old versions of jQuery, followed by old versions of the YUI Library.

Weak Passwords

Severity: High

Description

A weak password is short, common, a system default, or something that could be rapidly guessed by executing a brute force attack using a subset of all possible passwords, such as words in the dictionary, proper names, words based on the user name or common variations on these themes.

Impact

Weak passwords are the Achilles' heel of even the most well defended systems. A system is only as strong as it's weakest link, and when that link is an easily guessable password, that could potentially allow an attacker to gain access to restricted areas of a system and even escalate attacks further.

4% of sampled targets were found to be making use of weak passwords. This figure is up from last year's **1%**. Considering how trivial it is for an attacker to exploit weak or commonly used passwords, even on staging sites, it's baffling to see this figure rise.

Source Script Disclosure	Severity	High

Description

Source code often contains sensitive information, ranging from sensitive configuration information such as database credentials, or information on how the web application functions.

Impact

With disclosed source code, an attacker can leverage information obtained to escalate an attack by exploiting other vulnerabilities or misconfigurations discovered through the disclosed source code and/or configuration files.

10% of targets sampled were found to be vulnerable to source script disclosure vulnerabilities. This vulnerability affects the confidentiality of an application, it makes it possible for an attacker to gain access to sensitive files (possibly even files containing sensitive user data), configuration files and application source code. To such an extent, when one considers the possible ramifications associated with this vulnerability, it's certainly an area of web security that requires improvement.

WordPress related vulnerabilities	Severity	High

Description

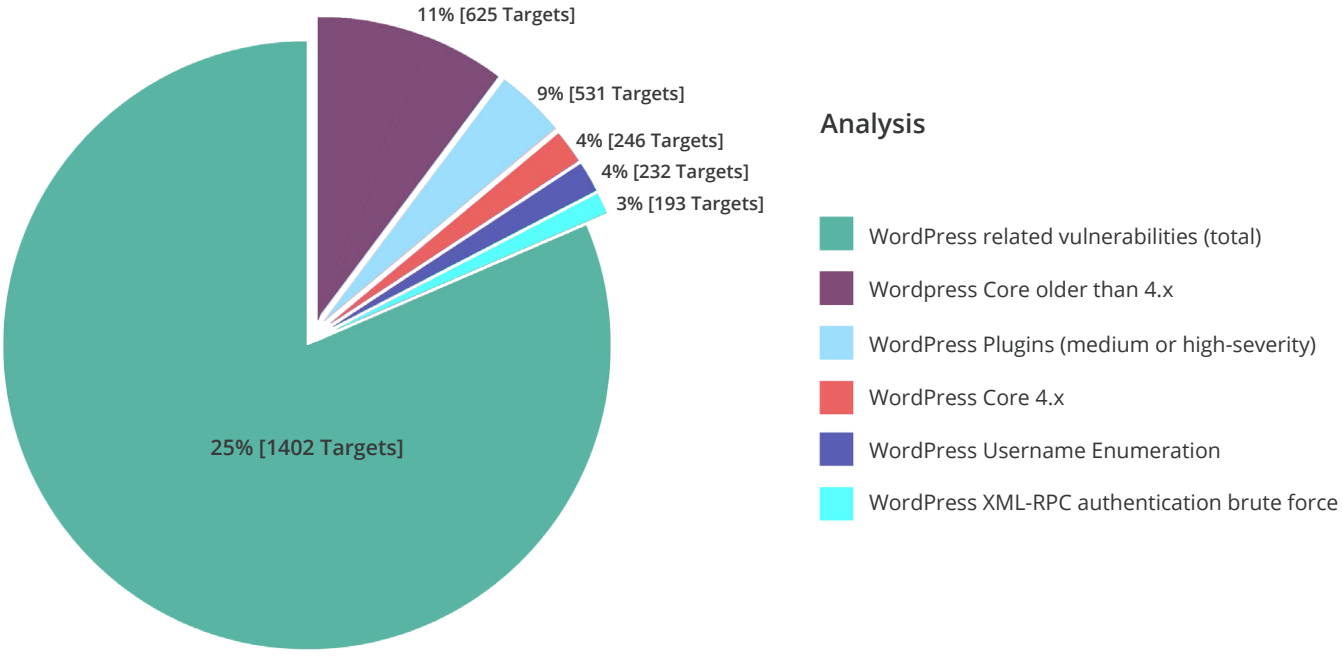
With 59% of all CMS-based websites making use of WordPress, its popularity means that it is prime target for attackers. While there are some inherent security weaknesses in WordPress' defaults, such as username enumeration, and XML-RPC authentication bruteforcing, the WordPress community strives to make security a priority, especially with automatic security updates turned on by default. Arguably the opposite can be said for the CMS' vibrant plugin and theme ecosystem.

WordPress security vulnerabilities that affect the WordPress core are relatively straight-forward to patch if you are running the latest version of WordPress. However, for the estimated 13% of sites on the internet running versions of the CMS that have been out of date for at least a year, upgrading may involve more effort due to incompatible old plugins and themes.

Plugins are the standard way to extend WordPress's core functionality. It's possible for **anyone** to write a plugin and distribute that plugin on the WordPress plugin repository. As a consequence, it is very common for plugins containing critical vulnerabilities to make their way into thousands of WordPress installations.

Impact

The impact a vulnerable WordPress installation and/or vulnerable plugins could have, will vary based on the kind of vulnerability in question. The vulnerability in question can range from cross-site scripting, all the way up to SQL injection and code execution.



[1] As of May 24th 2016 according to a W3Techs survey https://w3techs.com/technologies/details/cm-wordpress/all/all

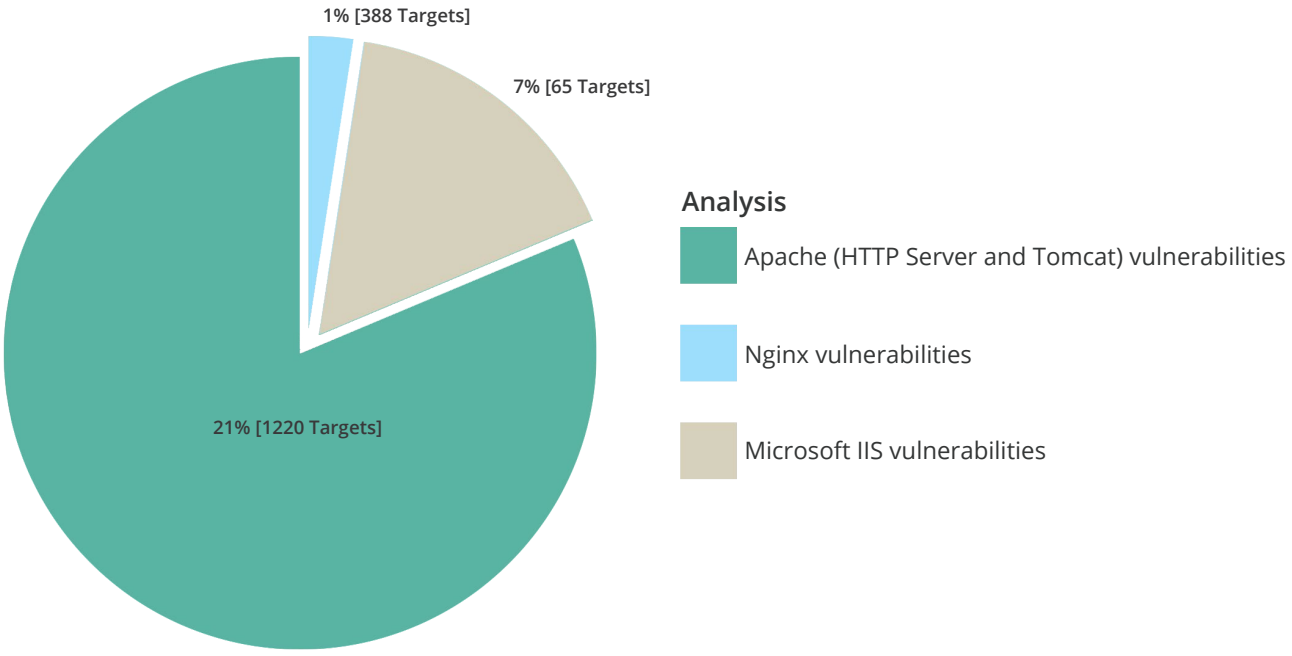
Web server related vulnerabilities	Severity	High

Description

Like all other software, web servers have bugs, some of which are security vulnerabilities. Running vulnerable versions of web server software is very far from ideal as it can easily lead to compromise, especially since, unlike most other software, web servers are **designed** to be publicly exposed.

Impact

The impact a web server vulnerability could have will depend specifically on the kind of vulnerability the web server is exposed to. A vulnerability in a web server can range all the way from information disclosure to a buffer overflow, which could allow an attacker to gain code execution.



Apache HTTP Server and Apache Tomcat vulnerabilities, together with their related vulnerable modules and misconfigurations accounted for a significant **21%** of all sampled targets. Amongst the most common vulnerabilities, were Apache HTTP Server remote denial of service (CVE-2011-3192) and Apache HTTP Server httpOnly cookie disclosure (CVE-2012-0053).

Microsoft IIS came in a distant second at **7%** of sampled targets. The large majority of IIS-related vulnerabilities can be attributed to the IIS tilde directory enumeration vulnerability, which allows an attacker to enumerate short names of files and directories potentially resulting in sensitive file disclosure.

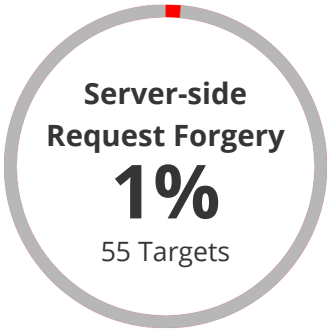
Nginx, only marginally showed up in this analysis with only **1%** of targets vulnerable to vulnerabilities concerning the increasingly-popular web server. The biggest contributor to this result was the Nginx SPDY heap buffer overflow vulnerability (CVE-2014-0133) in an old version of Nginx's SPDY (a precursor to the HTTP/2 protocol by Google) implementation, allowing a remote attacker to execute arbitrary code through a crafted request.

These results follow the same trends as web server popularity, it therefore stands to reason that Apache HTTP Server, being older and more prevalent than both Microsoft IIS and Nginx, accounts for a larger vulnerability and misconfiguration attack surface.

Server-side Request Forgery	Severity	High

Description

Server Side Request Forgery (SSRF) is a vulnerability which allows an attacker to create requests from a vulnerable server. SSRF attacks typically target systems on internal networks that sit behind firewalls, and are therefore not usually accessible from the outside world. SSRF makes it possible for an attacker to access these systems, as well as services running on the same server that is listening on the loopback interface (127.0.0.1/localhost).



Impact

Since SSRF allows an attacker to forge requests on behalf of the server, an attacker can scan and attack systems residing on the internal network that are not normally accessible externally, such as database services (MySQL, Elasticsearch, MongoDB...), caching services (Memcached, Redis...), and directory services (Microsoft Active Directory, OpenLDAP...). SSRF can be used to enumerate and attack services that are running on these hosts, and possibly even exploit host-based authentication services.

1% of sampled targets vulnerable to SSRF, indicates that SSRF is not as widespread as other high-severity vulnerabilities such as SQL injection, or even code execution. However, it's potential impact once the vulnerability is present is significant, especially in helping an attacker conduct detailed reconnaissance.

[2] As of May 2016 according to a Netcraft survey http://news.netcraft.com/archives/2016/05/26/may-2016-web-server-survey.html

Overflow vulnerabilities

Severity: High

Description

Overflow (such as buffer overflow, stack overflow and heap overflow) vulnerabilities exist when a program does not exercise proper bounds-checking, and as a result, overflows the buffer's boundary and overwrites adjacent memory locations while writing data to a buffer.

Impact

Overflow vulnerabilities can corrupt data, crash programs, or worse, cause the execution of malicious code. To such an extent, buffer overflows in software used to run web applications or network infrastructure (such as web servers, routers and mail servers) pose a serious security threat.

7% of sampled targets were found to be vulnerable to overflow vulnerabilities (buffer overflows, integer overflows, heap overflows, stack overflows...). The majority of this result can be attributed to the Easy File Management Web Server buffer overflow vulnerability, followed by the Nginx SPDY heap buffer overflow (CVE-2014-0133). Given the fact that most overflow vulnerabilities have stable exploits which can be easily used by attackers to gain code execution, this figure can certainly improve, especially because most of these vulnerabilities are just a missing patch or update.

Perimeter Network vulnerabilities

Severity: High

Description

Network perimeter vulnerabilities residing in network perimeter resources are typically results of configuration issues or vulnerabilities in devices such as routers, firewalls and other network appliances, or even services like web servers, mail servers and VPN gateways to name a few.

Because most organizations are now starting to move all, or parts of their infrastructure to the cloud the perimeter is no longer a clear-cut physical perimeter that surrounds an organization’s premises. On the contrary, the perimeter is changing to encompass everything from the corporate firewall to mail servers hosted in the cloud.

Impact

A misconfigured network device or service, or the presence of vulnerabilities in services on a network infrastructure, can cause havoc. An attacker often needs one small inlet into a network, and from then on escalating an attack is usually easy; either because the network is not properly segmented and/or not enough controls are in place to detect intruders within a network.

Analysis

13% of sampled targets were found to be vulnerable to Secure Shell (SSH) related vulnerabilities, predominantly concerning OpenSSH vulnerabilities. The most common vulnerabilities allow potential privilege escalation (CVE-2015-6564); allow local users to conduct impersonation attacks (CVE-2015-6563); make it easier for remote attackers to conduct brute-force attacks or cause a denial of service through CPU consumption (CVE-2015-5600); and vulnerabilities which make it easier for remote attackers to bypass intended access restrictions (CVE-2015-5352).

3% of sampled targets were found to be vulnerable to Mail related vulnerabilities. Unlike SSH and DNS related vulnerabilities, mail related vulnerabilities do not largely pertain to a specific software package, instead vulnerabilities are very much distributed amongst a variety of software packages and common misconfigurations to mail servers. Amongst the most common vulnerabilities, were tests against mail servers answering to VRFY and EXPN requests, which can confirm the existence of names of valid users, resulting in user enumeration; exported email CSV files and exposed sensitive mailbox files, which could lead to username and information disclosure; and open mail relay vulnerabilities.

3% of sampled targets were found to be vulnerable to Domain Name System (DNS) related vulnerabilities, with the majority of vulnerabilities centering around ISC BIND, the de facto, and most widely used DNS software on the Internet. The most common vulnerabilities all related to remote DoS vulnerabilities in ISC BIND (CVE-2015-5722, CVE-2015-8704, CVE-2015-5477, CVE-2015-8000).

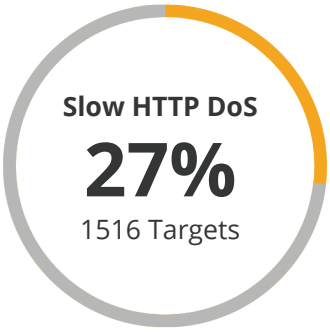
DoS related vulnerabilities	Severity	Medium

Description

Denial of Service (DoS) vulnerabilities affect the availability of a web application or another service, for instance DNS, SSH or SMTP daemons. DoS vulnerabilities therefore inhibit legitimate users from using the application normally since all resources end up being used by an application or a service to respond to an attacker's requests.

Impact

Because DoS attacks attempt to make a server, service or network resource unavailable to its intended users, it likely results in loss of business as well as increased resource usage, possibly resulting in extra infrastructural and data transfer costs.



43% of sampled targets were vulnerable to DoS attacks, with 27% of targets being vulnerable to a web-server specific DoS attack known as the Slow HTTP DoS attack. This vulnerability, also commonly referred to as Slowloris, is an attack which allows an attacker with a single machine to take down a web server with minimal bandwidth. The attack achieves this by making requests to a web server and never closing the connection, causing the server to run out of the maximum HTTP open connections allowed. As a result, once the attacker occupies all connections of the web server, requests made by legitimate users may not be fulfilled by the server until the attacker stops the attack.

Other common DoS vulnerabilities include the Apache HTTP Server remote denial of service vulnerability (CVE-2011-3192) and PHP DoS vulnerabilities (CVE-2015-7804, CVE-2015-7803)

▲ 5% Given the potential business impact of a DoS attack, defending against such attacks is important. Unfortunately, defending against DoS attacks is not easy and the truth is that no one is really immune to DoS attacks. However, mitigating against vulnerabilities like the Slow HTTP DoS attack, make it harder for attackers to achieve DoS through the use of simple vulnerabilities.

Cross-site Request Forgery	Severity	Medium

Description

Cross-Site Request Forgery (CSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF an attacker abuses the trust a website has with a victim's browser. An attacker could use CSRF to trick a victim into accessing a website hosted by the attacker, or clicking a URL containing malicious or unauthorized requests.

Impact

CSRF attacks leverage the identity and privileges of the victim when the forged request is being sent to the web server in order to perform actions desired by the attacker, such as change form submission details, and launch purchases or payments for the attacker or a third-party account.

Upon sending an HTTP request (legitimate or otherwise), the victim's browser will include the Cookie header. Cookies are typically used to store a user's session identifier in order to prevent the user from authenticating for each request, which would obviously be impractical. To such an extent, if the victim's authentication session is still valid (a browser window/tab does not necessarily need to be open), an attacker can leverage CSRF to launch any desired requests against the website, without the website being able to distinguish whether the requests are legitimate or not.

59% of sampled targets were reported to be susceptible to CSRF, or an HTML form without CSRF token. However, it is important to note that while it is possible to detect CSRF automatically, it is not possible to automatically determine if the alert is a real CSRF vulnerability or not. The reason for this is because not every HTML form necessarily has a sensitive action associated to it—an example of this would be an HTML form that submits a search query. Remember that the attacker has no way of retrieving any part of the response of what the victim requested, so even if the search box is in a restricted area, there isn't much an attacker can do by making a user submit a search query unintentionally.

Acunetix Vulnerability Scanner performs two different tests to attempt to detect CSRF vulnerabilities—a test for CSRF in POST requests, and another for HTML forms that do not have a CSRF token. Both alert types would need to be verified manually because of the aforementioned reason.

Host Header Injection

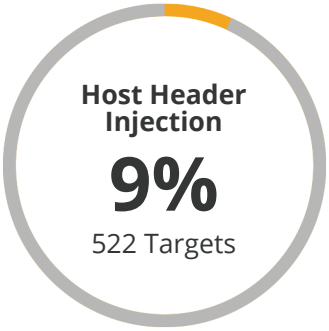
Severity: Medium

Description

A host header injection attack can occur as a result of a web application implicitly trusting the value inside the HTTP Host header and using it to generate everything from links, to import scripts and stylesheets on a page, and even generate password reset links.

Impact

Implicitly trusting the Host header is a bad idea, since it can be controlled by an attacker. This can lead to web-cache poisoning attacks to serve malicious content, as well as abusing alternative channels such as password reset emails getting sent to the attacker instead of an account's rightful user. The latter attack is known as password reset poisoning, and could allow an attacker to escalate an attack to account takeover, and possibly even escalate the attack even further if the account that is taken over is that of a high-privileged user.



9% of sampled targets were found to be vulnerable to host header injection, which is a **▼5%** reduction from last year's 14% figure. While a host header injection vulnerability bears a significant risk, it may not be as straight-forward to exploit, or it might require the attacker to 'get lucky' in order to take advantage of the vulnerability. Having said this, once again, this is a case of web developers placing implicit trust in user-controlled input, which could be easily avoided.

Directory Listing

Severity: Medium

Description

Directory Listing refers to a web server misconfiguration that could divulge sensitive information to an attacker. Directory Listing is a 'feature' that is enabled in some web servers by default which allows a user to view a list of files and directories hosted on the web server in an organized hierarchical view. An attacker can abuse this vulnerability by simply listing directories to find sensitive files.

Impact

Directory Listing can allow an attacker to escalate an attack by disclosing sensitive information or even configuration. For instance, an attacker can leverage a directory listing vulnerability to download source code and find other exploitable vulnerabilities in an application.

15% of targets sampled were found to be vulnerable to directory listing misconfigurations. While the fact that Apache HTTP Server enabling directory listing by default is certainly a contributor towards this misconfiguration, disabling directory listing is one of the very first, and elementary configurational changes one should make when first setting up a web server.

▼ 2% This figure dropped from last year's 17%, which is encouraging, however, clearly, more needs to be done to curb simple misconfigurations like directory listing—one of the most effective mitigations against this issue is secure default configurations. The Nginx web server for instance, does not automatically enable directory listing, instead the administrator would need to explicitly configure it to serve directory listing pages. This being said, system administrators should follow basic hardening guides when deploying services they may not be fully familiar with.

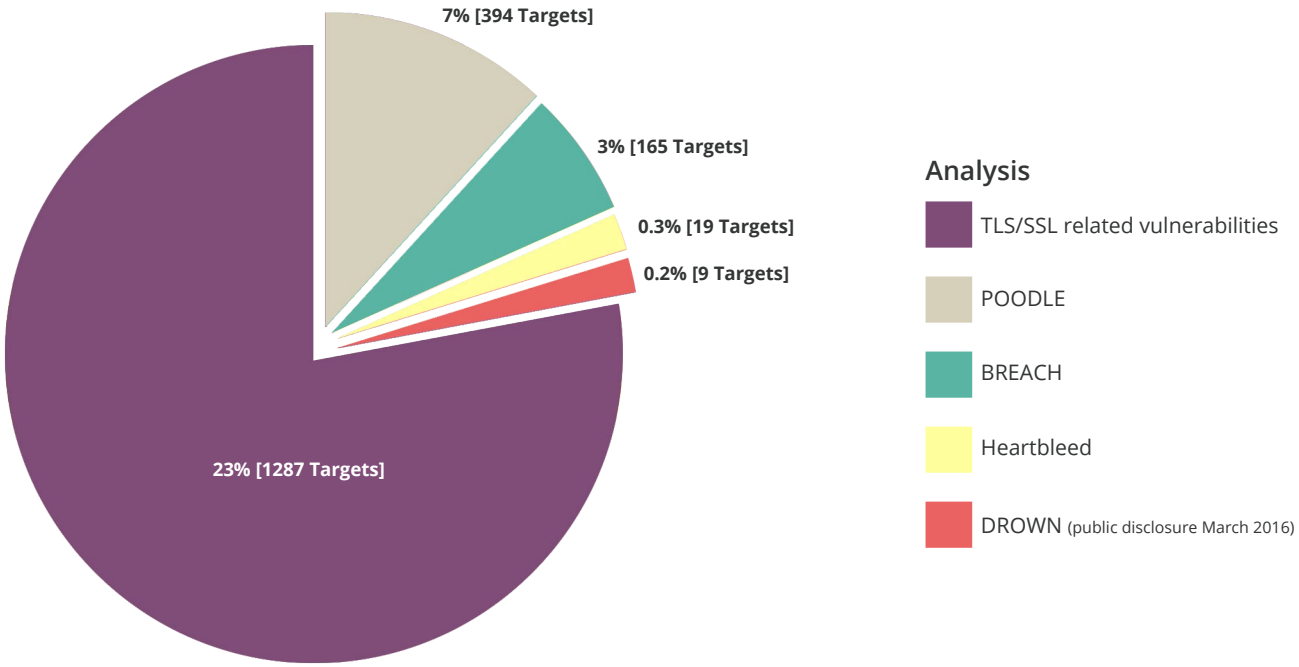
TLS/SSL related vulnerabilities	Severity	Medium

Description

Transport Layer Security (TLS) and its predecessor, Secure Socket Layer (SSL) are widely used protocols designed to secure the transfer of data between the client and the server through authentication, encryption and integrity.

Impact

TLS security is essential for websites and other services that rely the essential cryptographic protocols to allow browsing the web, using email, shopping online, and sending instant messages without third-parties being able to read or alter these communications.



23% of targets were found to have TLS/SSL issues. TLS/SSL issues were a hot news topic throughout 2014 and 2015, and while some vulnerabilities have been discovered and patched in libraries such as OpenSSL throughout 2016, these vulnerabilities were nowhere as serious, widespread, or hyped as the venerable TLS heartbeat read overrun vulnerability known as Heartbleed (CVE-2014-0160), of which, this year only 0.3% of targets were found to be vulnerable to.

Other TLS related vulnerabilities such as POODLE and BREACH though, have not seen the same decline as Heartbleed did. 7% of targets were found to be vulnerable to POODLE, and 3% of targets were found to be vulnerable to BREACH.

▼ 6% Furthermore, the DROWN vulnerability, which was publicly disclosed during March 2016, which is when the dataset was compiled, only affected 0.2%, down by 6% over last year at the time the dataset was compiled.

Conclusion

The analysis of the results obtained this year through Acunetix Online Vulnerability Scanner (OVS) clearly indicate that the web application attack vector is a major threat that organizations of all shapes and sizes around the world are facing - whether they are aware of it or not and they are compounding the problem by ramping up pressure on dev teams to deliver web projects ever faster.

This year, the majority of all websites worldwide have at least one high severity vulnerability, an increase of 9% since last year. While some specific vulnerabilities are in decline, the trend is worryingly upwards.

With vulnerabilities such as SQL injection, Cross-site Scripting and Code execution, the traditional 'patching' approach to mitigating the majority of 'traditional' network-layer vulnerabilities, is often not sufficient to defend an application against an attack. This is largely because web application vulnerabilities generally arise from poor design choices or oversights made during the development or deployment process. To make matters more complicated, no two web applications are the same—web applications include a lot of custom code, plugins, configuration and other customizations that are only used within that one application, but exposed to the world. It is therefore crucial to ensure that software is written and deployed securely in order for organizations to limit their exposure to risk.

Naturally, traditional abuse of network-layer vulnerabilities, malware and exploits are still pervasive methods of not only compromising machines, but also escalating attacks beyond the web application into other, potentially more sensitive areas of an organization's infrastructure—be that on premise or in the cloud. The heavy reliance on web technologies is an ideal target for attackers to exploit, and unfortunately, development teams are often up against tight deadlines, caught-up in complex engineering problems, and most are poorly equipped to assess the implications of insecure code within their applications.

With web application vulnerabilities increasingly posing serious threats to organizations' overall security posture, now is the time for organizations to make application-level security not only a priority, but a fundamental requirement.

With the industry-wide skills shortage and limited scalability that the conventional recruitment process offers, it's time for organizations to turn to alternative measures in order to establish a solid web application security baseline.

One such alternative is to **automate** the security testing process as much as possible. Naturally, automated testing, like any other security testing methodology, should not be viewed as a 'silver-bullet' solution, but rather, it should be seen as a highly cost-effective approach to establishing a baseline security posture. By leveraging automated security testing to uncover 'low-hanging-fruit', manual security testing, be that through a traditional penetration test, or through crowdsourced security testing platforms, is immediately more cost effective because penetration testers' focus is on finding hard-to-reach bugs that require human logic, hunches and intuition to discover.

Automated security testing provides a highly-scalable, cost-effective, ongoing security baseline all the way from the initial stages of the Software Development Lifecycle (SDLC) to Staging and Production environments.

With web application vulnerabilities increasingly posing serious threats to organizations' overall security posture, **now** is the time for organizations to make application-level security not only a priority, but a fundamental requirement.

About Acunetix Online Vulnerability Scanner

User-friendly and competitively priced, Acunetix Vulnerability Scanner fully interprets and scans websites, including HTML5 and JavaScript and detects a large number of vulnerabilities, including SQL Injection and Cross Site Scripting, eliminating false positives.

Acunetix not only excels in accuracy, speed and support of modern web technologies; but with over 3100 web application vulnerability tests, it provides the widest testing coverage, including the detection of out-of-band vulnerabilities such as Blind Cross-Site Scripting (BXSS), Server-Side Request Forgery (SSRF), XML External Entity Injection (XXE), Host Header Injection and more.

Acunetix also has the most advanced detection of WordPress vulnerabilities and a wide range of reports including HIPAA and PCI compliance.

Register for a free trial at:
http://www.acunetix.com/vulnerability-scanner/register-online-vulnerability-scanner/.

About Acunetix

Founded in 2005 to combat the alarming rise in web application attacks, Acunetix is the market leader, and a pioneer in automated web application security technology.

Acunetix products and technologies are depended on globally by individual pen-testers and consultants all the way to large organizations such as the **Pentagon, Nike, Disney, Adobe** and many more. For more information, visit www.acunetix.com/company.

acunetix

WHERE TO FIND US

Stay up to date with the latest web security news.

Website. **www.acunetix.com**
Acunetix Web Security Blog. www.acunetix.com/blog
Facebook. www.facebook.com/acunetix
Twitter. twitter.com/acunetix

CONTACT INFORMATION

Acunetix (Europe and ROW)
Tel. +44 (0) 330 202 0190
Fax. +44 (0) 30 202 0191
Email. sales@acunetix.com
Acunetix (USA)
Tel. (+1) 404 990 3280
Fax. (+1) 404 990 3279
Email. salesusa@acunetix.com