Web Application Vulnerability Report
2015
Contents

Introduction ............................................................................................................. 3
Methodology ........................................................................................................... 4
Results ................................................................................................................... 5
  Vulnerabilities by severity .................................................................................. 6
  Top Web Vulnerabilities ..................................................................................... 8
Cross-site Scripting ................................................................................................. 9
  Types of XSS ....................................................................................................... 9
    Stored XSS ..................................................................................................... 9
    Reflected XSS ............................................................................................... 10
    DOM-based XSS ............................................................................................ 10
SQL Injection ......................................................................................................... 11
  Blind SQL Injection ........................................................................................... 12
TLS/SSL Related Vulnerabilities ............................................................................ 13
Superbugs 2014 - 2015 ......................................................................................... 14
Directory Listing .................................................................................................... 16
Host Header Attack ................................................................................................. 16
Vulnerable JS Libraries ........................................................................................... 16
WordPress Vulnerabilities ....................................................................................... 17
Web technology specific vulnerabilities ............................................................... 18
Vulnerabilities that could lead to DoS ................................................................. 19
Vulnerabilities by Web Server ............................................................................... 20
Network Vulnerabilities ......................................................................................... 21
Conclusion ............................................................................................................. 23
About Acunetix Vulnerability Scanner *(Online version)* ..................................... 24
About Acunetix ....................................................................................................... 24
Introduction

Welcome to Acunetix Web Application Vulnerability Report 2015. A year after the release of the online version of our vulnerability scanner in March 2014, Acunetix have aggregated the findings of over **15,000 scans** performed on **1.9 million files** over the past **12 months** with some interesting results. The report details the most common vulnerabilities found, how often they occurred and which bugs our users have been susceptible to.

Overall, our findings confirm that web application vulnerabilities are increasingly posing serious threats to organizations' overall security posture, such as data loss or alteration, system down-time, loss of reputation and severe fines from the regulator, amongst others. Website security should be a priority in any organization, but remains the most overlooked aspect of securing the enterprise. Hackers continue to concentrate their efforts on web-based applications since they often have direct access to back-end data such as customer databases.

Recent high-profile attacks have included **Sony Pictures**, **Target** and **Anthem Insurance** among many others. The nature of cyber attacks is also diversifying as criminals target not only financial data but personal data for use in identity theft and confidential intelligence to carry out cyber espionage.

This report offers an insight into the types of security issues most websites are vulnerable to. We feel it's important to share this data as it can help to inform website owners of their biggest risks of attack. While web application security is our focus, Acunetix also provides the ability to run full free network vulnerability scans via our online service.
Methodology

The data was gathered from Acunetix Online Vulnerability Scanner - the Acunetix cloud-based solution. All data was collected over a period of one year ending March 2015. The evaluation scans on the Acunetix test websites were not included.

For the purpose of this report statistics are taken on a random sample of 5,500 users who have successfully scanned at least one scan target, out of a possible 18,000 subscribers.
Results

Our findings show that both web applications and perimeter servers are vulnerable to high and medium security vulnerabilities. Given the level of IT security awareness, the degree to which web application and perimeter network devices remain vulnerable to attack is somewhat alarming. Nearly half of the web applications scanned contained a high security vulnerability such as XSS or SQL Injection, while over 4 out of 5 web applications were affected by a medium security vulnerability.

The chart shows that administrators are better geared to protect against Network Vulnerabilities, however the stats are not comforting at all. **10% of the servers scanned were found to be vulnerable to high security risks**, and **50% had a Medium security vulnerability**. Keeping in mind most of these servers are perimeter servers, having a network vulnerability on these internet-facing servers could spell disaster, as this could easily lead to server compromise and possibly be escalated further.

**Nearly half of the web applications scanned contained a high security vulnerability such as XSS or SQL Injection.**

<table>
<thead>
<tr>
<th>NETWORK</th>
<th>WEB</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

- **9%** High Risk Servers Vulnerability
- **50%** Medium Risk Servers Vulnerability
- **46%** High Risk Servers Vulnerability
- **87%** Medium Risk Servers Vulnerability

1500 distinct types of vulnerabilities were discovered

11 distinct types of WordPress vulnerabilities discovered
Vulnerabilities by Severity

Severity is a metric for classifying the level of seriousness 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 we consider the impact of each vulnerability severity level.

<table>
<thead>
<tr>
<th>High Severity</th>
<th>Medium Severity</th>
<th>Low Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>An attacker can easily exploit such vulnerabilities to compromise the integrity and availability of the target application, gain access to backend systems and databases, as well as deface the target site and trick users into phishing attacks.</td>
<td>An attacker can exploit such vulnerabilities caused by server misconfiguration and site-coding flaws, which facilitate server disruption and intrusion. Medium severity vulnerabilities could also be used to escalate an attack by exploiting known vulnerabilities in disclosed software components.</td>
<td>An attacker can identify sensitive information derived from lack of encryption of data traffic, or directory path disclosures and may be able to use this information to escalate an attack and find other vulnerabilities.</td>
</tr>
</tbody>
</table>

The fact that half of the web applications scanned contained a high security vulnerability such as XSS or SQL injection and 4 out of 5 contained a medium security vulnerability such as Directory Listing or POODLE, demonstrates that over 50% of the organizations scanned would fail at PCI Compliance.

As stated in the PCI Security Standards, “Forensic analyses of cardholder data compromises have shown that web applications are frequently the initial point of attack upon cardholder data, through SQL injection in particular”. “The intent of Requirement 6.6 is to ensure web applications exposed to the public Internet are protected against the most common types of malicious input”. “
For PCI Compliance standards, the ASV Program Guide states that “To demonstrate compliance, a scan must not contain high or medium severity vulnerabilities, or any vulnerability that indicates features or configurations that are in violation of PCI DSS.” Any vulnerability with a CVSS base score of 4.0 or higher will result in a non-compliant scan.

Acunetix contains a report set up to present any risks of non-compliance, structured as per the PCI specification.

---

1. PCI Security Standards Council, Information Supplement: Requirement 6.6 Code Reviews and Application Firewalls Clarified

2. Payment Card Industry (PCI) Approved Security Vendors Program Guide
   > https://www.pcisecuritystandards.org/pdfs/asv_program_guide_v1.0.pdf
Top Web Vulnerabilities

Cross-site Scripting (XSS) and Denial of Service (DoS) vulnerabilities top our list with a significant 38% of websites being vulnerable to each of these attacks. Following closely at 28% are SSL related vulnerabilities such as HeartBleed and POODLE, and SQL Injection (SQLi) at 27% of the sites scanned by Acunetix OVS. This graph shows that new security vulnerabilities, such as the super bugs discovered last year and earlier this year are already nearly as common as the older vulnerabilities such as XSS and SQLi, that have been around for decades.

Figure 2.

Servers exposed to commonly detected vulnerabilities.
Cross-site Scripting

Cross-site Scripting (XSS) is a term that refers to client-side code injection. While XSS can be taken advantage of within VBScript, ActiveX and Flash (although now considered legacy or even obsolete), unquestionably, the most widely abused is JavaScript - primarily because JavaScript is fundamental to most browsing experiences.

XSS can be used in a range of ways to cause serious problems

One of the traditional (and dangerous) uses of XSS is the ability for an attacker to steal session cookies allowing an attacker to impersonate a victim. It has been used to wreak havoc on social networks, spread malware, phish for credentials and even used in conjunction with social engineering techniques to escalate to more damaging attacks.

Types of XSS

Cross-site Scripting can be classified into three major categories - Stored XSS, Reflected XSS and DOM-based XSS.

Stored XSS

The most damaging of the three is Stored (Persistent) XSS. Stored XSS attacks involves 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). The victim would then browse the website, and inadvertently execute the malicious script once the page is viewed in his browser.

Figure 3.

Identified Cross-site Scripting (XSS) vulnerabilities.
Reflected XSS

The second, and by far most common type of XSS is Reflected XSS. In Reflected XSS, the attacker’s payload script has to be part of the request which is sent to the web server and reflected back in such a way that the HTTP response includes the payload from the HTTP request. Using Phishing emails and other social engineering techniques, the attacker lures the victim to inadvertently make a request to the server which contains the XSS payload and ends-up executing the script that gets reflected and executed inside the browser. Since Reflected XSS isn’t a persistent attack, the attacker needs to deliver the payload to each victim - social networks are often conveniently used for the dissemination of Reflected XSS attacks.

DOM-based XSS

DOM-based XSS is an advanced type of XSS attack which is made possible when the web application’s client side scripts write user provided data to the Document Object Model (DOM). The data is subsequently read from the DOM by the web application and outputted to the browser. If the data is incorrectly handled, an attacker can inject a payload, which will be stored as part of the DOM and executed when the data is read back from the DOM.

The most dangerous part of DOM-based XSS is that the attack is often a client-side attack, and the attacker’s payload is never sent to the server. This makes it even more difficult to detect for Web Application Firewalls (WAFs) and security engineers analysing the server’s logs since they will never even see the attack.

Among various objects that make up the DOM, there are some objects in particular which an attacker can manipulate in order to generate the XSS condition. Such objects include the URL (document.URL), the part of the URL behind the hash (location.hash) and the Referrer (document.referrer).

Our findings show a whopping 95% of XSS vulnerabilities involved Reflected Cross-site scripting, with only 5% being made up of DOM-based and Stored XSS.
SQL Injection

Since its first public discussions appearing around 1998, SQL Injection is one of the oldest and most prevalent of software bugs that is still being actively exploited today. The latest Open Web Application Security Project (OWASP) Top 10 still lists Injection as the most dangerous class of 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 inject SQL commands into, say a login form or a search field, allowing them to gain unauthorized access to data held in the backend database.

SQL Injection is possible when inputs are either incorrectly filtered for escape characters, or user input is not properly validated, in such a way that an attacker could manipulate SQL queries. Such weaknesses in an application’s design provide attackers with the ability to craft malicious requests to the web application, effectively enabling them to run SQL statements and query the database directly.

The quickest and most basic way to check for SQL Injection is to enter a single quote (‘) or a double quote (”) (commonly referred to as a ‘tick’) in the field or parameter that the attacker is trying to launch an attack from. This will cause a syntax error in the SQL statement, leading the SQL interpreter to produce an error, which most of the time will be displayed within the web application, clearly indicating that the field is vulnerable to SQL Injection.

The attacker will then most likely craft particular requests to cause the database to disclose information about itself within the error responses. The attacker can use this information to identify the type of database being used (this process is typically referred to as fingerprinting), build the database schema and retrieve data from different tables in the database.

Web server administrators quickly realised that showing errors to the general public is not a wise thing to do, so they started suppressing detailed error messages. This is obviously a flawed solution, since it does not address the underlying problem – user input can still be parsed by the SQL Interpreter as part of an SQL statement.


SQL Injection accounted for over 25% of vulnerabilities detected.
**Blind SQL Injection**

SQL injection is still possible when the results of the injection are not visible to the attacker. This is referred to as Blind SQL Injection. Unlike its Error-based counterpart, pages vulnerable to Blind SQLi do not display data within the response from the server, however the page will display differently depending on the results of a logical statement injected into the SQL query. The two techniques used to achieve a Blind SQLi attack are – **Boolean-based Blind SQL Injection** and **Time-based Blind SQL Injection**.

While SQL injection is mostly used to steal data from the database, an SQLi vulnerability can easily be escalated further if the permissions on the database are incorrectly configured. For example, the attacker can inject a query that causes some tables to be deleted from the database, effectively causing a DOS attack. The attacker can also take over the server hosting the database using Remote Code Execution (RCE) by constructing SQL queries that execute code on the database server.

Our findings show that out of the 1455 SQL injection vulnerabilities detected, 829 scanned websites were found to be vulnerable to Blind SQL Injection.

![Figure 4. Identified SQL injection vulnerabilities.](image-url)
TLS/SSL Related Vulnerabilities

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 and encryption and integrity.

2014 and so far, 2015 have been bad years for TLS/SSL. Starting with Heartbleed in April of 2014, continuing with the Padding Oracle On Downgraded Legacy Encryption (POODLE) vulnerability in September of 2014 and the Factoring RSA Export Keys (FREAK) vulnerability discovered in March 2015.

The below figure shows the most commonly found TLS/SSL related vulnerabilities. 37% of websites vulnerable to SSL vulnerabilities were found to be using what are considered weak ciphers. POODLE account for a staggering 23% of TLS/SSL related vulnerabilities. This is probably caused by the web administrator’s reluctance to stop supporting older browsers.

Unsurprisingly, given the massive media coverage, and a simple fix in the form of a patch; the Heartbleed vulnerability was only found on 3% of websites. Older TLS/SSL related vulnerabilities such as CRIME are still around, but affect less than 1% of websites. The FREAK vulnerability was made public only a few days prior to the data for this report being compiled. The data gathered did not include enough instances of this vulnerability to make it into the report under its own category, instead, it is included as part of the OTHER category in the chart above.

The Payment Card Industry Data Security Standard (PCI DSS) has been recently updated to include the recommendation of implementing TLS in favor of SSL. TLS is the successor of SSL, bringing with it improvements to the protocol. At the time of writing, there are no known security vulnerabilities in the latest version of the TLS protocol. Considering that 29% of websites scanned were found to be vulnerable to SSL-specific vulnerabilities, the move to TLS configurations that make use of strong ciphers is becoming increasingly important.
Superbugs 2014 - 2015

Security ‘Superbugs’ were a prominent topic throughout 2014 and so far throughout 2015. Most superbugs achieved a great deal of publicity and discussion, particularly in online forums and in the mainstream media. Most of these bugs were given catchy nicknames or acronyms and in some cases, even logos; leaving system administrators in a race against time to patch systems before attackers started to exploit them just hours later. Acunetix was able to automatically detect these Superbugs in less than 24 hours after public disclosure.

Heartbleed (CVE-2014-0160), dubbed as the bug that “Broke the Internet” 4. The Heartbleed Bug is a high-severity vulnerability OpenSSL, a widely used cryptographic software library. The bug allows attackers to steal encrypted data as a result of a leak of data in memory from the server to the client without leaving a trace of the attack.

Shellshock (CVE-2014-6271), was arguably the most severe Superbug of 2014 assigned the highest CVSS score of 10. Shellshock is actually a series of bugs in the massively popular Bourne Again Shell (Bash). This bug is very easy to exploit allowing remote attackers to execute arbitrary code via a crafted environment.

Padding Oracle On Downgraded Legacy Encryption (POODLE) (CVE-2014-3566). The POODLE vulnerability takes advantage of a padding oracle vulnerability during the fallback to SSL 3.0 from a more secure protocol through a man-in-the-middle attack. While, the POODLE vulnerability is less severe than other 2014 superbugs, our findings indicate that it is very rampant, probably due to the wide support for the legacy SSL 3.0 protocol.

The GHOST vulnerability (CVE-2015-0235) consists of a buffer-overflow in a function of the Linux glibc library. It allows attackers to remotely execute commands on a target system. Some web applications could be affected by this flaw, however, the potential for attackers to exploit the bug within web applications is low.

The Factoring attack on RSA-EXPORT Keys (FREAK) vulnerability (CVE-2015-0204) was the first SSL bug for 2015. The FREAK vulnerability allows attackers to intercept HTTPS connections between vulnerable clients and servers and force them to use ‘export-grade’ cryptography, which can then be decrypted or altered.

MS15-034 (CVE-2015-034) is a security bulletin released by Microsoft to address a vulnerability in HTTP.sys that could allow remote code execution on affected systems. This vulnerability was made public after the data for this report was compiled and is therefore not represented within this report.

Such wide-spread bugs will continue to be exploited in the years to come, and this is clearly shown in our scan results, which are still detecting bugs (like BREACH) that had been discovered nearly 2 years ago.
Directory Listing

Directory Listing refers to a 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 folders hosted on the server in an organized hierarchical view. An attacker can abuse this vulnerability by simply listing directories to find sensitive files.

Directory Listing can lead to other attacks. For instance, an attacker can leverage a directory listing vulnerability to download source code and find other exploitable vulnerabilities in an application.

Host Header Attack

A Host Header attack occurs when an attacker has the ability to control functionality within web applications that are implicitly trusting the HTTP Host header value. Some applications make use of the host header to generate links, import scripts and even generate password resets links. Since the HTTP Host header can be controlled by an attacker, an attacker can exploit this vulnerability through web-cache poisoning attacks and by abusing alternative channels such as password reset emails.

Vulnerable JS Libraries

Most websites and web applications frequently leverage one or more JavaScript libraries to enhance the user experience of the site, as well as to build core functionality of the web application. Running vulnerable versions of JavaScript libraries such as jQuery, jQuery UI, YUI, PrototypeJS, EmberJS and Dojo are inherently at risk of Cross-site Scripting vulnerabilities present in the vulnerable versions of those frameworks.

Vulnerable JS Libraries were identified on 664 web applications, which accounted for 11.99% of all servers scanned.
With 48% of Technorati’s Top 100 blogs being managed with WordPress and an impressive 74.6 million sites depending on it, it’s safe to say that WordPress is top dog when it comes to Content Management Systems. It is known for its ease of use and scalability, but its popularity means that it is increasingly making the headlines for the wrong reasons - its lack of security. There are 2 major areas that affect WordPress security - vulnerabilities that affect the WordPress core and others that affect specific plugins or themes. In both cases, exploitation of these vulnerabilities can lead to full WordPress installation compromise.

The top two WordPress vulnerabilities detected were 144 instances of WordPress Username Enumeration and 85 instances of XML-RPC Authentication Brute force.

By default, WordPress allows enumeration of usernames. This gives attackers a head start when attacking WordPress installations since an attacker would have the necessary information to launch a password dictionary attack against the enumerated usernames. To make the situation worse, WordPress provides an XML-RPC interface that uses HTTP for transport protocol and XML for encoding which is designed to enable remote posting and other advanced features. An attacker who used username enumeration to fish-out usernames can abuse this interface to brute force authentication credentials using API calls such as wp.getUsersBlogs. Such an attack is faster and stealthier than attacking the main WordPress login screen.

---

Web Technology Specific Vulnerabilities

As cited in the W3Techs 6 PHP web applications are more popular when compared to ASP web applications. This preference was also visible in our scan data - indicated by the black line on the graph below. The downside is that PHP web applications are generally less secure. This suggests that ASP.NET facilitates the development of secure web applications. On the other hand, PHP developers have to manually insert safeguards or rely on frameworks more often to produce the same level of secure code.

With the exception of the POODLE vulnerability, all the other major vulnerabilities are more common on PHP web applications. POODLE might be more common on IIS, IIS v7 and prior versions that have SSLv3 enabled by default, which requires multiple changes in the Windows Registry to disable.

Vulnerabilities that could lead to DoS

38% of the servers scanned contained vulnerabilities that make the site open to a Denial of Service (DoS) that can be so crippling to modern Ecommerce and online operations. The most common of these vulnerabilities is the Slow HTTP DoS Attack, often referred to simply as Slowloris. This type of attack makes multiple requests to the server causing the server to run out of the permitted HTTP open connections.

The next most common DoS vulnerability is specific to Apache and is caused by the way multiple overlapping ranges are handled by the Apache server. Discovered in 2011, CVE-2011-3192 is known to cause a significant amount of CPU and memory usage on the server.

Another common DoS vulnerability affects the TCP protocol when the OS is configured to use a large TCP Window Size, making it easier for a remote attacker to guess the sequence number and cause a DoS. Over a decade after it was first reported, CVE-2004-0230 is one of the older vulnerabilities that is making the rounds.

One of the less common DoS vulnerabilities affects mail servers which include an Anti-virus plugin. These can be affected by the DoS if the Anti-virus cannot correctly handle nested ZIP files, or ZIP files which include thousands of files.

DDoS, i.e Distributed Denial of Service attacks are on the increase. This is no big news, judging by attacks such as the ones which took down the Playstation Network (PSN) and Xbox Live during the 2014 Christmas period; even the average news-viewing technophobe might have been able to predict this. In light of this, the fact that 38% of scans found vulnerabilities which can lead to DoS attacks is significant.

If these type of attacks continue to increase as anticipated, measures to mitigate risk of becoming a victim are essential.

Figure 9. Identified DOS vulnerabilities.
Vulnerabilities by Web Server

We also analysed the vulnerabilities which are specific to the 3 most popular web servers - Apache, IIS and nginx.

We scanned 2716 Apache web servers finding 18,266 vulnerabilities; 1111 IIS servers finding 5029 vulnerabilities; and 303 nginx servers finding 1391 vulnerabilities. Most vulnerabilities were found in Apache web servers, this might give the assumption that Apache is more vulnerable when compared to the other servers, however one must remember that Apache is still the most popular web server owning 38.39% of the market share as listed on netcraft.  

Surprisingly however, we detected considerably more vulnerabilities in web applications running on IIS when compared to nginx. The Web Server Survey conducted by Netcraft in April 2015 lists nginx as being more popular than IIS, with nginx being used by 21.22% of the top million sites, compared with 12.21% using IIS.

![Web servers scanned and vulnerabilities found.](image-url)
Network Vulnerabilities

In this section, we provide an overview of the protocol specific vulnerabilities that we have identified.

Topping the list are SSH related vulnerabilities, and the most common is CVE-2012-0814, which allows remote authenticated users to obtain sensitive information from the server. Mail Systems are still often configured incorrectly to be open relays. This might be caused by the administrator forgetting to disable the mail server which was included as part of the default OS installation. We also discovered server-specific vulnerabilities, most of which have been discovered over a year ago, such as CVE-2011-3208, which is a stack-based buffer overflow vulnerability affecting Cyrus IMAP server.

One of the most common DNS vulnerabilities identified can be used for DNS Amplification attacks, which are often used for DoS related attacks. Other DNS vulnerabilities include DNS spoofing, DNS cache poisoning and buffer overflows in specific DNS servers.

Similarly, the vulnerabilities identified for the FTP protocol are just as dangerous. These include buffer overflows, vulnerabilities that lead to DoS, security bypass and mis-configured FTP server that allow writable directories for anonymous users. Other protocols were also affected, but to a lesser extent.

Figure 11. Identified Web Server Vulnerabilities.

Acunetix Web Application Vulnerability Report 2015
Acunetix OVS includes network vulnerability scanning, which can be used to identify network vulnerabilities in the perimeter. This service is being offered entirely free without any limitation on time by registering at:

› http://www.acunetix.com/free-network-security-scanner

**Acunetix Vulnerability Scanner (Online version) network scanning features**

- Detects over 35,000 vulnerabilities, including HeartBleed, POODLE and ShellShock
- Detects misconfigurations and vulnerabilities in OS, server software, network services, and protocols
- Assesses security of detected device (routers, hardware firewalls, switches and printers)
- Scans for trojans, backdoors, rootkits, and other malware that can be detected remotely
- Tests for weak passwords on FTP, IMAP, SQL servers, POP3, Socks, SSH, Telnet
- Checks for DNS server vulnerabilities such as Open Zone Transfer, Open Recursion and Cache Poisoning
- Tests FTP access such as anonymous access potential and a list of writable FTP directories
- Checks for badly configured Proxy Servers, weak SNMP Community Strings, weak SSL ciphers and many other security weaknesses.
Conclusion

Having surveyed over 5,500 organisations, these findings are extremely troubling. In the race to produce user-friendly interfaces and customer-centred apps, modern companies are leaving their precious data wide open to cyber criminals. One look at the news headlines shows cyber-attacks are all too common. With nearly half of web apps containing a high security vulnerability such as XSS or SQL Injection, it’s just like leaving your wallet or unlocked phone lying around in a public place. It’s more a question of how long it takes, rather than if at all, before you are compromised.

But far from reducing the risks, companies are taking a too casual approach, and vulnerabilities are worse than ever – perhaps due to the complexities of today’s IT infrastructure. CSOs and CISOs need to take action now as this problem of unsecured apps is only getting worse as new technologies are adopted. The advice is simple: organizations’ ongoing security strategies must include web app and network vulnerabilities to avoid their data falling into the wrong hands.

Furthermore, web application vulnerabilities are increasingly more prevalent than network vulnerabilities. With cyber attacks constantly on the increase, this finding confirms that many organizations are still failing to develop secure software and patch vulnerabilities, some of which have a low barrier of entry, so remaining vulnerable to common attacks.

Cross-site Scripting and SQL Injection, the two most well-known methods of web app attack, remain two of the most common vulnerabilities detected in a large percentage of the scans. Most concerning is that many scans found the main superbugs of 2014 have not been patched, especially POODLE.

Other encryption-related vulnerabilities were also high, with 29% of scans detecting SSL protocol vulnerabilities such as weak ciphers, certificate issues and use of old versions of SSL. This also ties in with the lack of priority given to POODLE, which is still a vulnerability present in 13% of the applications scanned. We strongly urge users to switch to TLS as soon as possible, especially if PCI compliance is required.

Another area for concern is the susceptibility to DoS attacks, 38% of the applications scanned had a vulnerability that can lead to a DoS attack. With DoS attacks having doubled in number in 2014, priority needs to be given to preventing such an attack.

Another cause for concern is that network vulnerabilities on the perimeter are still being discovered. 4% of the servers scanned had email related issues, which also included open email relays, while 1.3% had DNS related vulnerabilities.

Security is not simply a case of reacting to each new risk as it appears, companies need to be proactive too. By focusing on security throughout the development lifecycle many of the risks can be mitigated there and then, through using best practice design and coding, meaning an application then simply requires regular scanning and patching where necessary. As we have seen from the bugs old and new which continue to be exploited, patching is very important, as are regular updates to encryption protocols, servers and software.

Don’t forget, once vulnerabilities are identified they cannot be ignored – companies must take action by using best practice techniques to close holes and limit risk.
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 beats competing products in many areas; including speed, the strongest support of the latest types of web application development architectures such as JavaScript, the lowest number of false positives and the ability to access restricted areas with ease. 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

Acunetix is the market leader in web application security technology, founded to combat the alarming rise in web attacks. Its products and technologies are the result of a decade of work by a team of highly experienced security developers. Acunetix’ customers include the U.S. Army, KPMG, Adidas and Fujitsu. More information can be found at www.acunetix.com