Invicti

The Invicti™ AppSec Indicator

Spring 2022 Edition: Worrisome Vulnerability Trends in the Race to Innovation

01

REPORT OVERVIEW

The digital world has accelerated through a transformation these past few years, but one truth remains: **nobody is safe from subpar security strategies.**

In our hyper-focus on the shift to the cloud and enabling hybrid workforces for success, it's never been clearer that web application security (AppSec) needs to stay top of mind for organizations large and small. Otherwise, they risk severe consequences around customer and company data exposure.

Are severe vulnerabilities getting any scarcer?

The short answer is no.

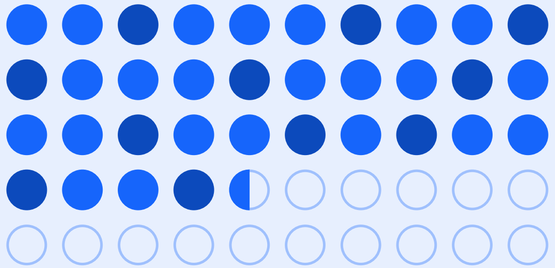
Without proper preparation, big breaches and dangerous vulnerabilities continue to expose our blind spots left and right. Just look at the whirlwind impacts of Log4Shell, which blindsided businesses around the globe as they scrambled to check whether or not they were vulnerable to remote code execution (RCE) attacks in the widely-used Log4j library. We've all refocused on this problem, but we're not paying close enough attention to why these vulnerabilities keep appearing, what we need to prioritize when we're overloaded, and which tools we're missing from our arsenal.

Log4Shell made us do a double-take at the software supply chain. It shook us by the shoulders as a stark reminder that things aren't improving and that a robust and multi-layered approach to security is required to avoid being vulnerable to lingering threats.

But with time-strapped teams and competing priorities, how do we get there without compromising? **By baking security into every step of the software development process.** The Open Web Application Security Project (OWASP) points to the concept of secure design to set the tone, the definition of which encourages organizations to build a culture that is always on top of threats and design their code in a way that prevents known attacks.

For those adjusting to the new reality of hybrid remote work, *it's no easy feat:*

69%



of organizations found that difficulties around security hygiene and posture management have increased over the last two years.[1]

Lack of bandwidth for boots-on-the-ground teams, trouble getting buy-in from the top, and a massive talent shortage in IT security – all of these issues contribute to the ongoing problem. And so, success in modern application security requires more than just a thoughtful combination of skills and tools. You need clear top-down initiatives that the entire organization can follow and an eye on the trends for direct-impact vulnerabilities.

We made a bold statement in the last edition of the Invicti AppSec Indicator, and we'll echo that here: ***security is everyone's job now.*** As you're upgrading legacy tools, streamlining scans, and fixing disjointed teams, you must keep an eye on trends in vulnerabilities and potential threats to stay ready, stay agile, and open new doors for innovation.

[1] ESG Research Report, Security Hygiene and Posture Management, January 2022

2021

OUR BIGGEST DATA SET TO DATE

170

HOURS SPENT ANALYZING REPORT DATA

939

CUSTOMER INSTANCES STUDIED

282,914

DIRECT-IMPACT VULNERABILITIES FOUND

23,630,985,830

SECURITY CHECKS PERFORMED

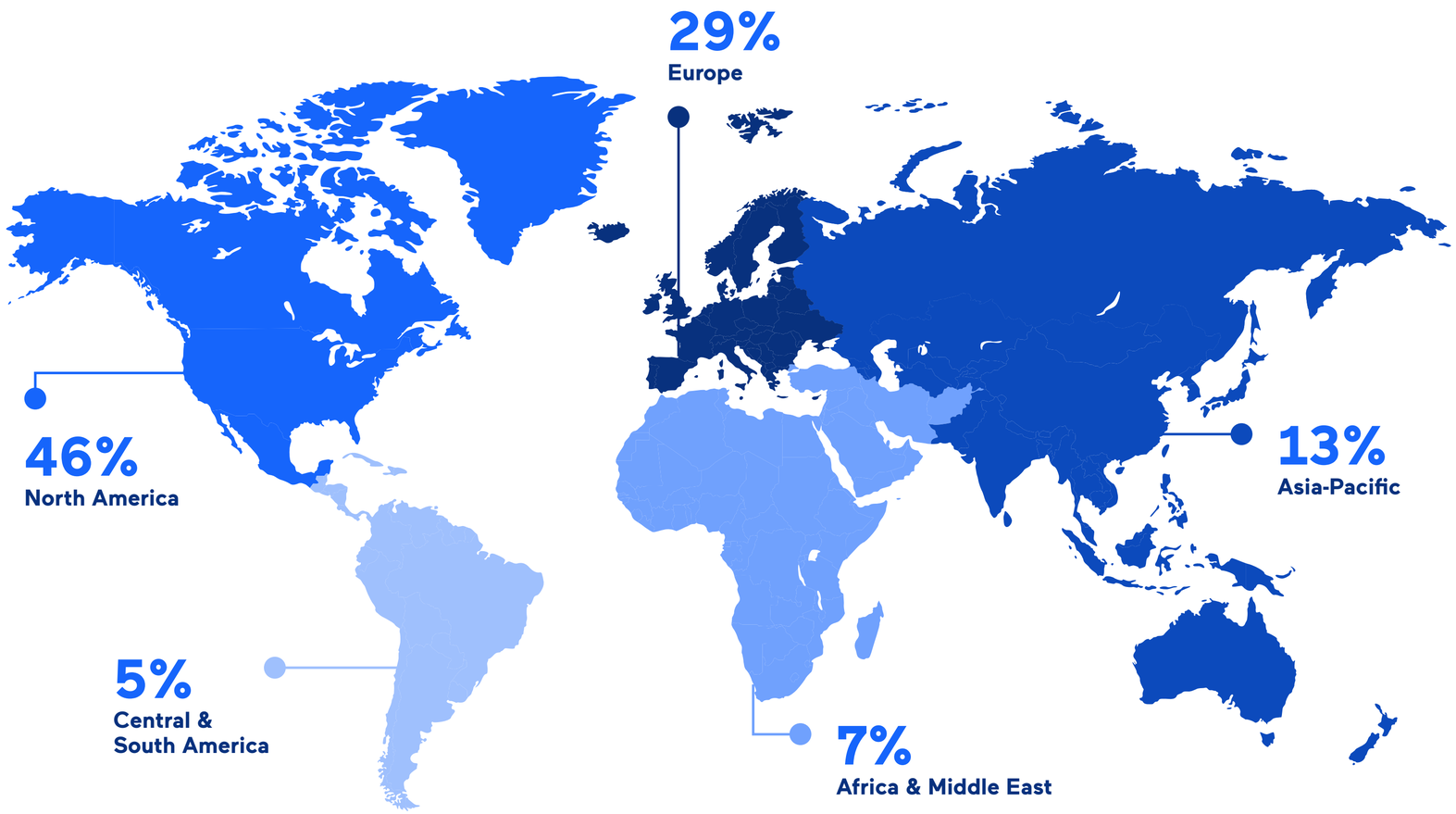
We're keeping an eye on industry trends, so this report surveys direct-impact flaws identified in our customers' environments to uncover year-over-year trends in the precedence of these vulnerabilities. This year we leaned on a deeper and richer data set, analyzing aggregated usage data from 939 global Invicti customers, which gave us a holistic view of vulnerability trends from automated scan results across regions.

With our growing customer base and the increased recognition of the importance of scanning everything, 2021 was our biggest year to date. We blew our previous scan records out of the water and our products ran more scans than ever: in 2021 alone, that added up to a whopping 23,630,985,830 security checks attempting to find and confirm vulnerabilities. And that's just for our cloud products, with many more checks done by on-site installations.

We wanted deeper insight into the state of web application security, wondering to what degree common vulnerabilities were popping up in scans year-over-year and contributing to ongoing risk.

AppSec by region

Here we can see the global makeup of our customers, which provides us with a more comprehensive view into vulnerability trends by region.



Invicti's AppSec platform in action

Automation reduces manual tasks by integrating multiple specialized features and functions into one platform, so that developers can build sophisticated software while coordinating work cross-functionally.

1

DISCOVER & CRAWL:

Automated scanning tools use advanced discovery services and crawling technologies to discover, authenticate, and analyze every asset, identifying potential entry points for attackers.

2

SCAN:

Comprehensive scan results provide detailed information for each flaw, including proof for the majority of direct-impact vulnerabilities.

3

REPORT:

Your scanning tool should provide extensive reporting with dashboards, predefined reports, and configurable templates to help monitor security posture, satisfy compliance, and ensure company-wide clarity.

4

REMEDIATE:

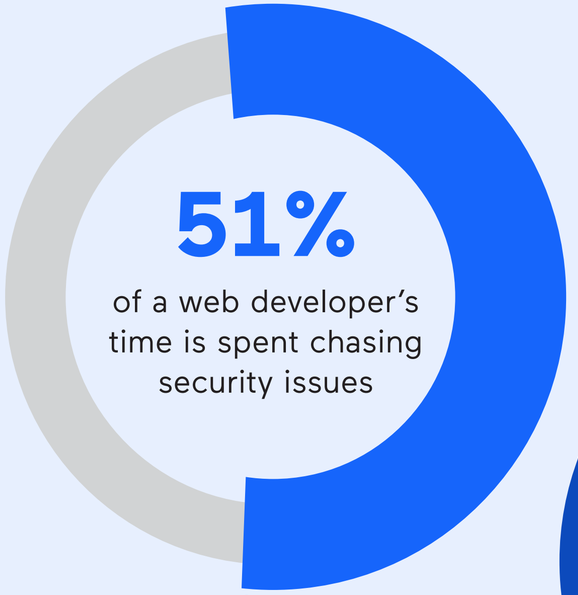
With a combination of dynamic, interactive, and composition analysis (DAST + IAST + SCA), no asset is left behind. Deeper coverage through blended scanning helps maximize confidence in scan results and pinpoint vulnerabilities to facilitate efficient remediation.

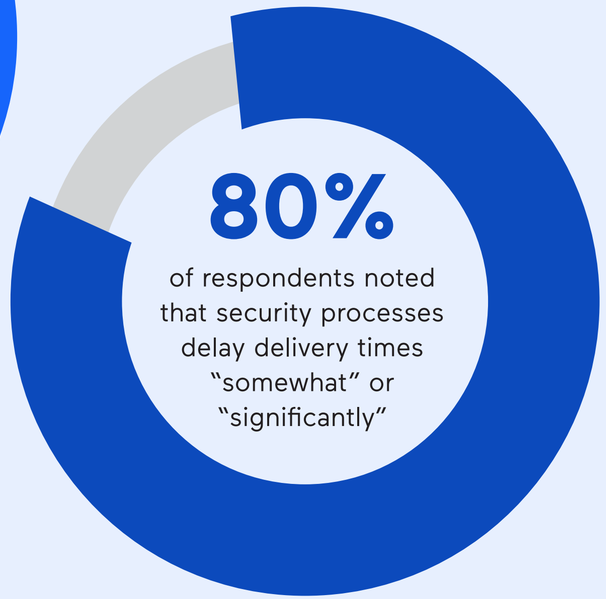
02

YEAR-OVER-YEAR TRENDS

The Fall 2021 version of the Invicti AppSec Indicator fielded a survey focused on the working relationship between security and development teams, and what we can do to **make lives easier on both sides of the aisle.**

The numbers were staggering: on average, 51% of a web developer's time is spent chasing security issues, and 80% of respondents noted that security processes delay delivery times "somewhat" or "significantly." We also uncovered that 1 in 3 security issues identified in scanning made it to production without being flagged in the development or test stages. *That's a big problem that can lead to serious security risk and debt over time.*





1 in 3

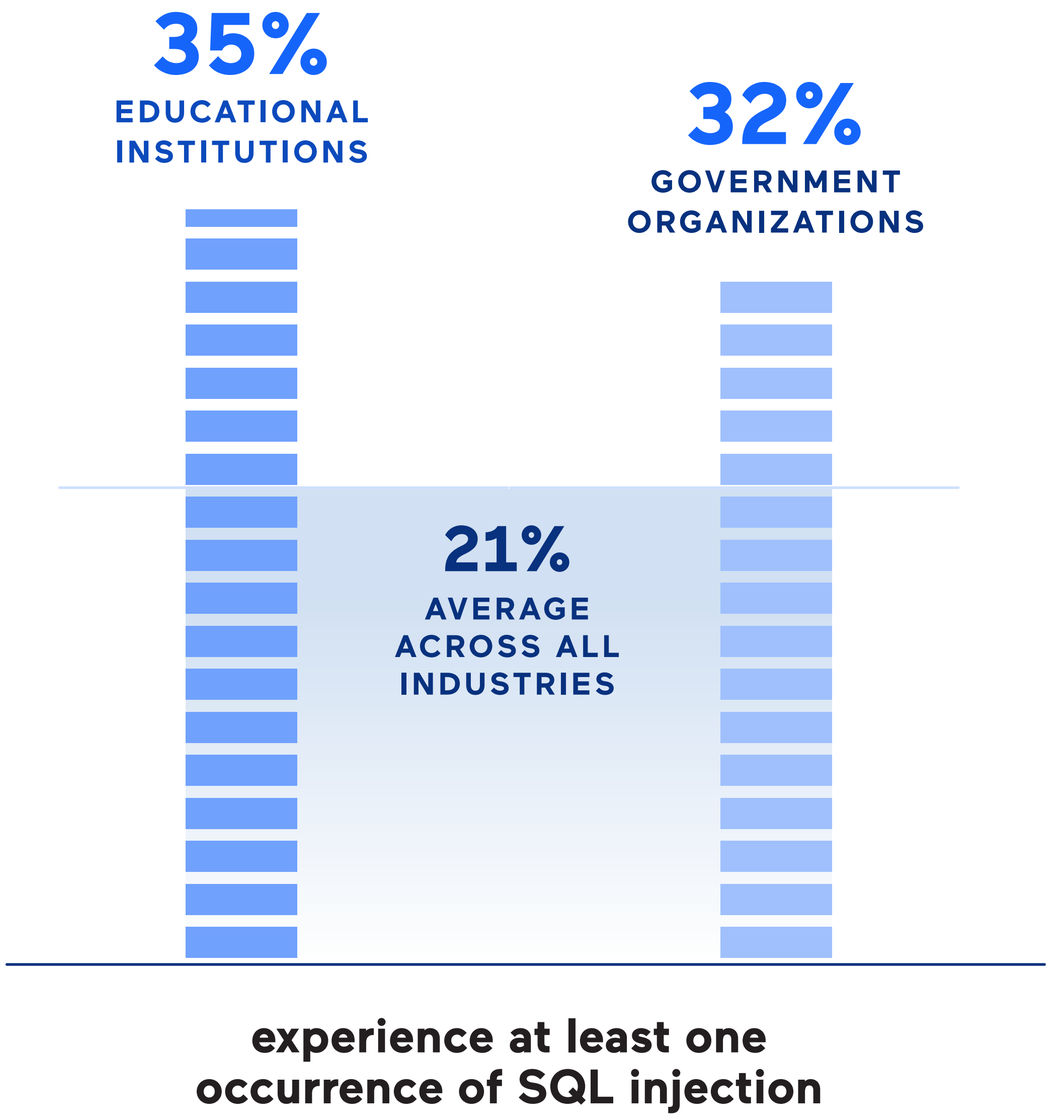
security issues under remediation make it to production without getting flagged in the test or development stages

READ MORE

The Invicti AppSec Indicator – Fall 2021 Edition ▸

Legacy technologies linked to SQLi in government and education sectors

In our research for this edition, we were able to glean some insights into vulnerability patterns in two sectors for 2022; specifically, education and government. The data suggests that these environments see more vulnerabilities that are typically attributed to older, legacy web technologies often found in industries like government and education. There are some concerning numbers for SQL injections, particularly: in our data, 35% of educational institutions and 32% of government organizations experience at least one occurrence of SQL injection, signaling legacy code that needs modernizing and developer knowledge gaps that impede progress. The damage that SQLi can do is far-reaching in these sectors. For example, government databases that are breached likely contain deeply personal information with details that attackers can use to assume identities to carry out illegal activities.

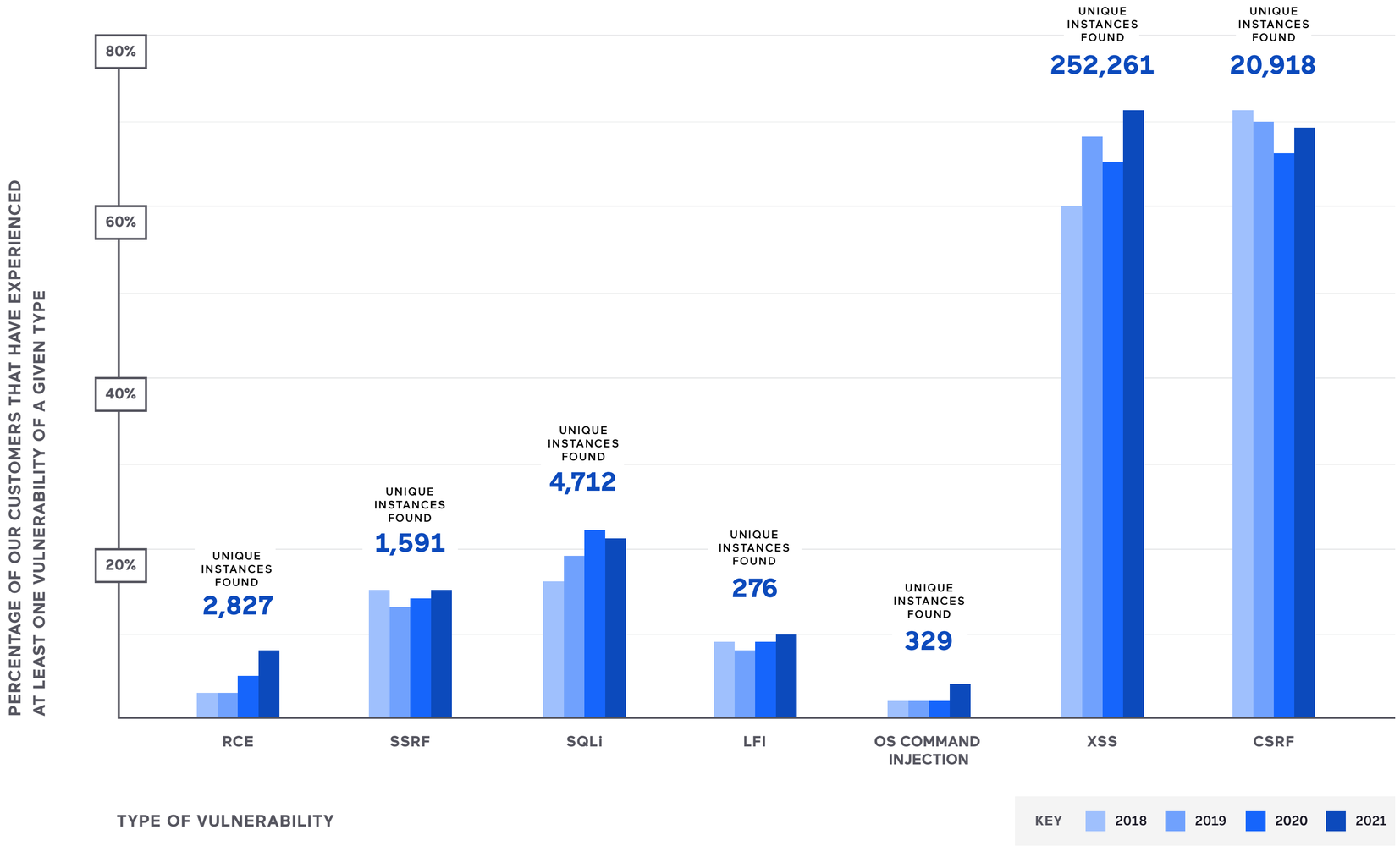


Well-understood vulnerabilities remain far too commonplace. The top offenders (RCE, XSS, and SQLi) typically lead to some pretty serious consequences – for example, compromised back-end data, hijacked sessions, or forced actions on behalf of other users and services. And even **if a threat seems like a softball, it's often just the beginning of a multi-stage event where common web vulnerabilities open the door for other attacks to follow.** This can lead to full control of the back-end server and even the possibility of attackers compromising connected systems.

Let's dig a little deeper. →

Our top high-severity vulnerabilities and their trends for the last four years

This four-year snapshot shows the percentage of customers by license that encountered at least one vulnerability of the given type in the given year.



With severe web vulnerabilities on a steady rise, it's clear that there's still work to do in preventing and remediating these security flaws. The graph on the previous page shows the number of times we've helped companies discover RCE, SSRF, SQLi, LFI, and OS command injection, which are hovering around the same number year after year or increasing slightly.

Remote code execution, for example, has seen a steady and worrisome increase since 2018, jumping 5% in frequency. Cross-site scripting – one of the most common flaws – saw a slight improvement in 2020, only to bounce back with a vengeance and a 6% jump. That tells us these flaws are still slipping under the radar or falling victim to time-strapped teams, and things just aren't getting better. As we dive into each of these direct-impact vulnerabilities with prevention and mitigation steps, take note of just how many flaws can lead to sensitive data loss, costly ransomware, and industry-shaking incidents like Log4Shell.

SINCE 2020

▲ 6%

increase in cross-site scripting

SINCE 2018

▲ 5%

increase in frequency of remote code execution

Well-known vulnerability classes have still **held strong or increased** in frequency over the past four years.

Same old, same old:

2020 vs 2021

trends show that some vulnerabilities are found at the same frequency as the previous year.

1%

Local file inclusion (LFI)

UNIQUE INSTANCES FOUND **276**

2%

OS command injection

UNIQUE INSTANCES FOUND **329**

1%

Server-side request forgery (SSRF)

UNIQUE INSTANCES FOUND **1,591**

3%

Remote code execution (RCE)

UNIQUE INSTANCES FOUND **2,827**

1%

SQL injection (SQLi)

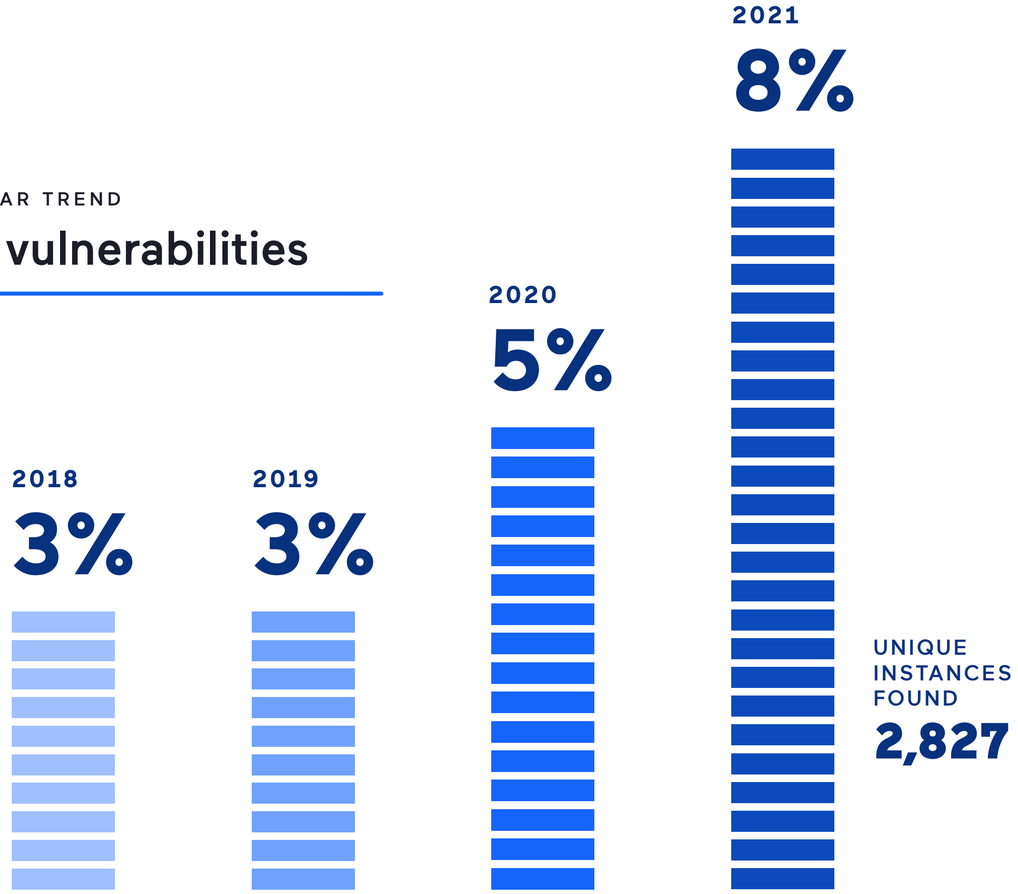
UNIQUE INSTANCES FOUND **4,712**

03

COMMON VULNERABILITIES: A CLOSER LOOK

FOUR-YEAR TREND

RCE vulnerabilities



Remote code execution (RCE), also referred to as code injection or remote code evaluation, allows malicious hackers to supply harmful code to an application and execute it. If a web application is vulnerable to RCE, a remote attacker can trick it into doing practically anything that the application's programming language allows. This can include installing a web shell to provide convenient remote access to extract sensitive data, deploying malware, or probing and attacking other systems on internal networks.

Because remote code execution can have such far-reaching consequences, it is considered the ultimate goal of any cyber attacker. Having even a single RCE vulnerability in a production environment puts you at risk of a complete system compromise. In the 2021 edition of OWASP's list of Top 10 web security risks, RCE and other injection flaws are at #3, grouped under Injection and down from #1 in 2017.

Learn more about RCE ▸

2021's Log4Shell in Apache Log4j 2.x is an example of a dangerous RCE flaw. It impacts multiple Log4j versions and is incredibly easy to exploit, allowing bad actors to take full control to steal data, install ransomware, and server-hop. Popular apps and services were initially vulnerable as organizations rushed to mitigate the problem.

Pro tip

The rising prevalence of individual RCE vulnerabilities may be related to more applications being deployed in the form of multiple containerized components, thus multiplying the potential attack surfaces for RCE.

Server-side request forgery (SSRF) happens when an attacker is able to masquerade as a web application when communicating with other websites, applications, or systems. If an application that accesses an external web resource is vulnerable to SSRF, a malicious user can alter its intended behavior and trick it into accessing a different resource.

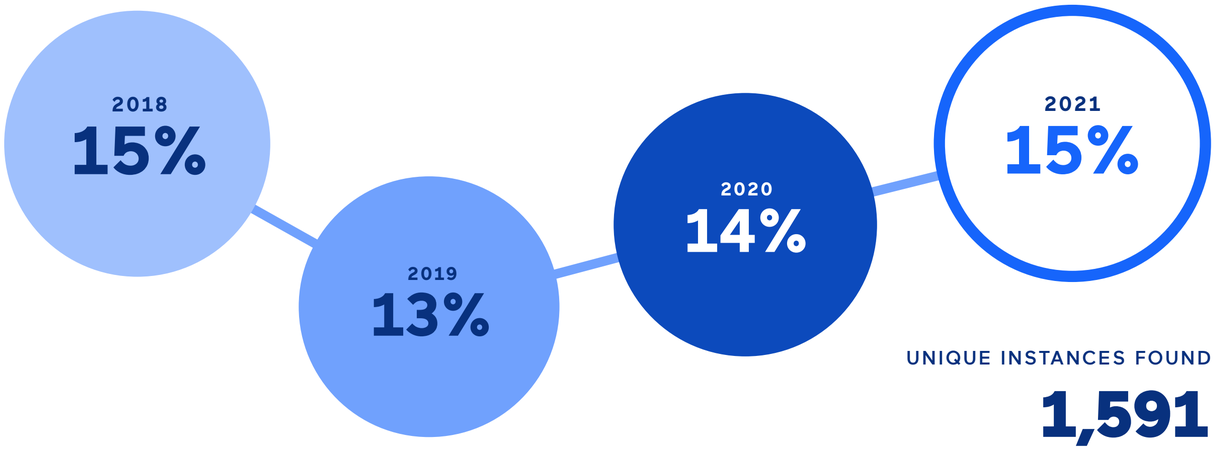
The most serious SSRF scenario is when web-based attackers are able to access systems that should be off-limits to them. This can happen when a vulnerable application is supposed to fetch an external web resource (say, a news feed) but accepts malicious data that causes it to fetch a file from its own private environment, such as an internal network or cloud bucket. SSRF vulnerabilities can lead to direct data breaches, but they are far more dangerous when used as gateways to other attacks. SSRF is the only flaw with a dedicated category in OWASP's Top 10 2021 ranking.

Pro tip

With many web applications now relying on implicitly trusted APIs for internal communication, SSRF attacks can be especially dangerous, as they can allow attackers to obtain unauthenticated API access via a trusted server.

FOUR-YEAR TREND

SSRF vulnerabilities



UNIQUE INSTANCES FOUND

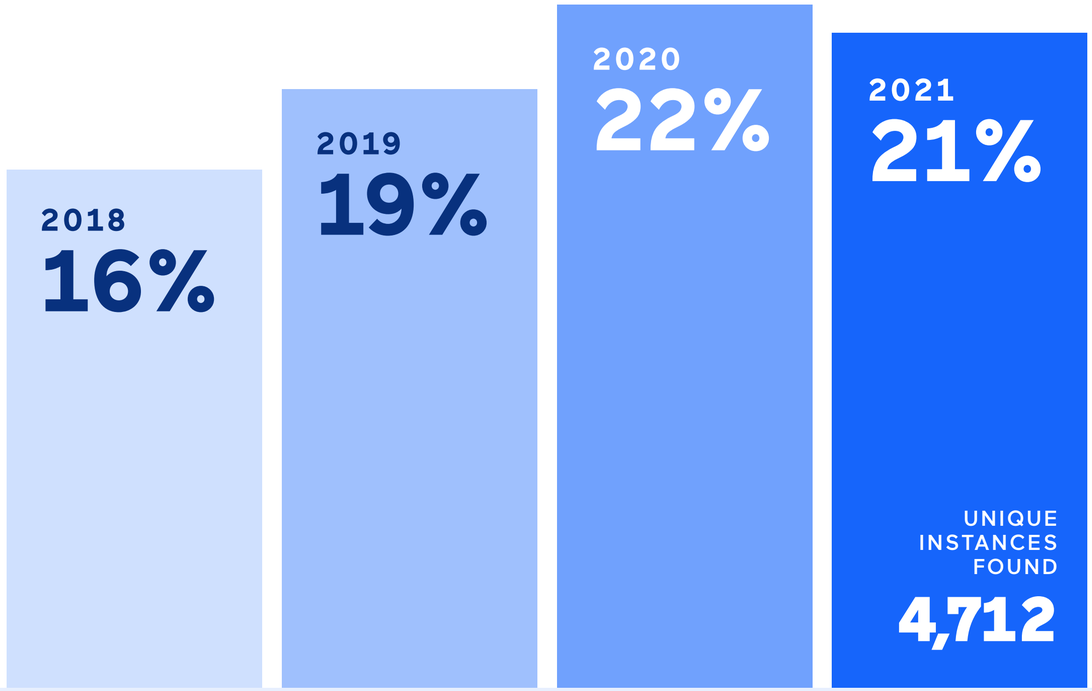
1,591

Learn more about SSRF ▸

In 2019, a misconfigured web application firewall allowed Paige Thompson to acquire AWS access keys by exploiting SSRF. Thompson gained access to data from approximately 106 million Capital One customers in the United States and Canada. Although this incident put the world on high alert, SSRF is bouncing back at a steady climb.

FOUR-YEAR TREND

SQLi vulnerabilities



Learn more about SQLi ▸

In 2019, a SQLi vulnerability resulted in the theft of tax data for millions of Bulgarian citizens. Records dating back ten years were exposed on underground forums, including full names, income information, personal identification numbers (EGN), social benefits, and more.

SQL injection (SQLi) allows malicious actors to modify or replace queries that an application sends to its database. Most web applications use a database that accepts queries in the SQL language and responds by returning the requested data or performing specified database operations. If an application builds its database queries from unsanitized inputs that an attacker can control, it is vulnerable to SQL injection.

Depending on the application and the database configuration, a successful SQLi attack can lead to unauthorized access, compromised credentials, leaked company data, or even complete data loss if an attacker deletes the database tables. Couple that with the associated costly downtime, and SQLi is an expensive problem. Despite being one of the oldest web vulnerabilities with well-known mitigation methods, SQL injection still occurs in modern web applications.

Pro tip

Technically speaking, SQL injection is very easy to prevent with modern web languages and frameworks, so its persistent presence is a reminder that application security starts with educating developers to correctly and consistently use the available safeguards.

Local file inclusion (LFI) allows bad actors to trick a web application into exposing or running specific files on the server. It is closely related to directory traversal (also known as path traversal), but while directory traversal lets attackers navigate to files that are outside the web application's main directory, LFI applies to files within that directory. If a web application accepts unvalidated user input and uses it when loading a file, it may be vulnerable to LFI. A typical instance would involve putting file names directly in the URL, where malicious users can manually modify them.

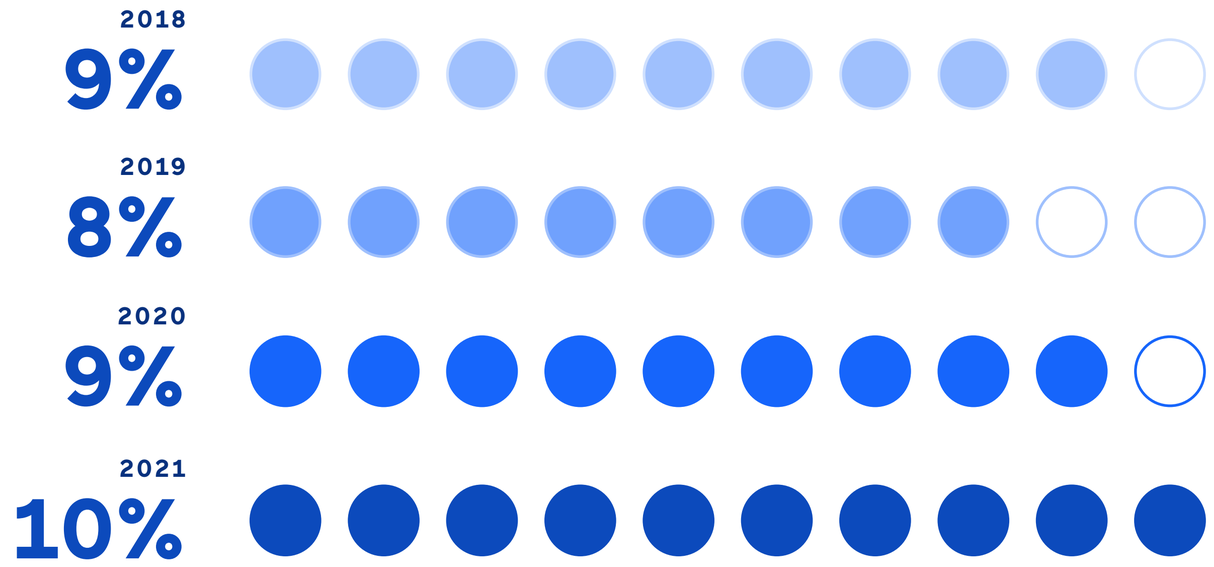
Local file inclusion usually results in information disclosure, allowing attackers to download files that should otherwise be inaccessible to them. For example, if you have confidential data on your web server in files that only authorized application users should have access to, an LFI vulnerability may allow attackers to directly display or download those files. In combination with other vulnerabilities, LFI can facilitate attacks such as cross-site scripting, command injection, or even remote code execution.

Pro tip

Historically, LFI vulnerabilities have primarily been associated with PHP, so their persistence in our reports can safely be correlated with the continued popularity of PHP-based applications and plugins – most notably, WordPress.

FOUR-YEAR TREND

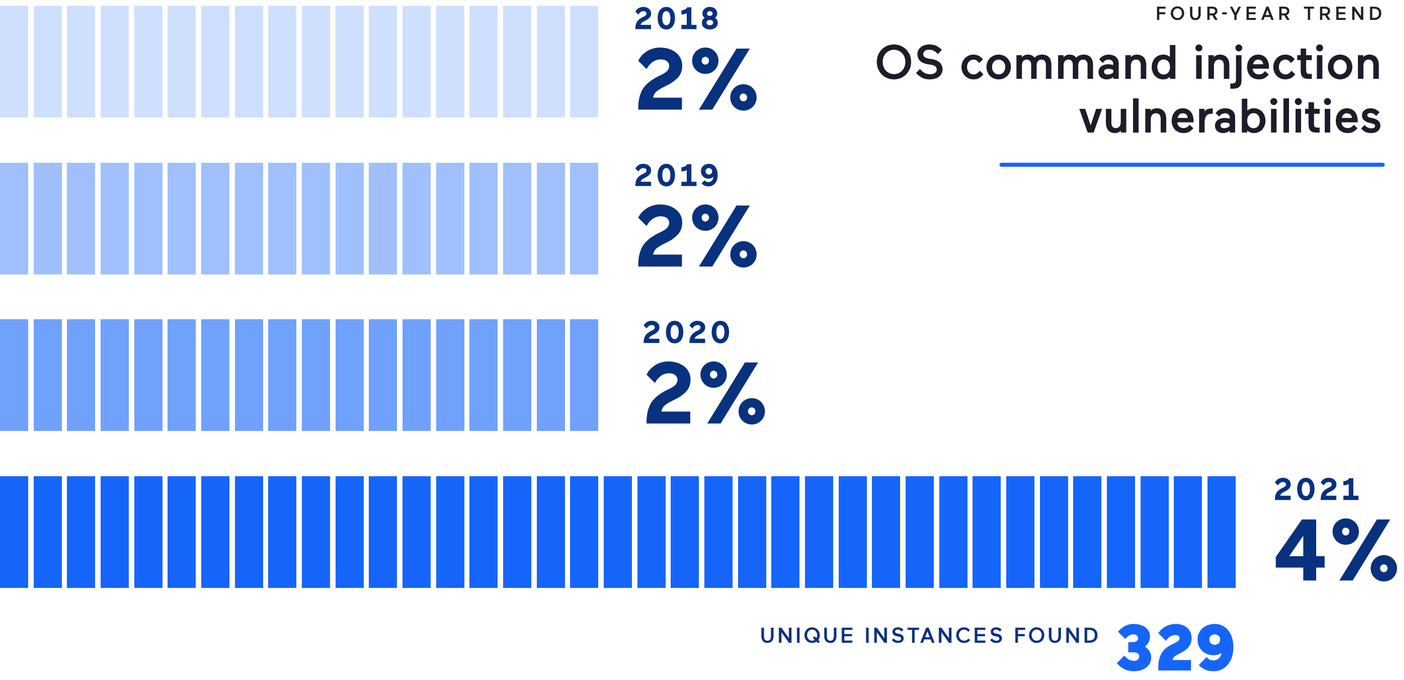
LFI vulnerabilities



UNIQUE INSTANCES FOUND **276**

Learn more about LFI ▸

After a slight dip in 2019, LFI is back in business. Hopefully, history won't repeat itself: a 2016 attack on Adult Friend Finder showed us just how dangerous LFI vulnerabilities can be. Bad actors were able to access information for 412 million user accounts, including emails, passwords, and confidential data about relationship status.



FOUR-YEAR TREND

OS command injection vulnerabilities

UNIQUE INSTANCES FOUND **329**

Operating system command injection, or simply command injection, lets attackers execute system commands through a web application. A website or application is vulnerable to command injection if it calls operating system commands and incorporates unvalidated user input when preparing these commands. A vulnerable application might insert user input directly into a command parameter, allowing malicious hackers to enter a specially-crafted attack string that executes additional commands in the application's local operating system.

OS command injection is extremely dangerous. It can allow attackers to take control of the web server system, extract sensitive data, install a web shell for remote access, deploy malware, and perform further attacks from the compromised system.

Learn more about OS command injection ▸

For an often-overlooked flaw, OS command injection saw an alarming jump from 2020 to 2021. If you want to know how bad it can be, just look at Shellshock, a widely-known vulnerability that enabled malicious actors to inject OS commands and take control. Within hours of vulnerability disclosure in 2014, attackers exploited the flaw by creating computer botnets that performed millions of DDoS attacks worldwide.

Pro tip

Similar to RCE, the rise in command injection flaws may be related to the growing popularity of modular, containerized deployments where each service or microservice potentially presents a separate injection target, though impact will vary widely depending on the service.

Cross-site scripting (XSS) allows attackers to inject malicious JavaScript code and execute it in the user's browser. There are many variants of XSS, corresponding to the many ways that unsanitized user input can make its way into JavaScript executed by the browser.

Though considered a low-impact vulnerability by many developers, XSS is dangerous because it can open the way to sensitive data disclosure, session hijacking, redirects to malicious sites, malware installation, and social engineering attacks. For example, threat actors may send out legitimate-looking links that start with your known and trusted domain name but end with disguised malicious script code that exploits a cross-site scripting vulnerability in your website. In effect, anyone clicking that link to your website will be attacked and assume that the attack came from your site.

Pro tip

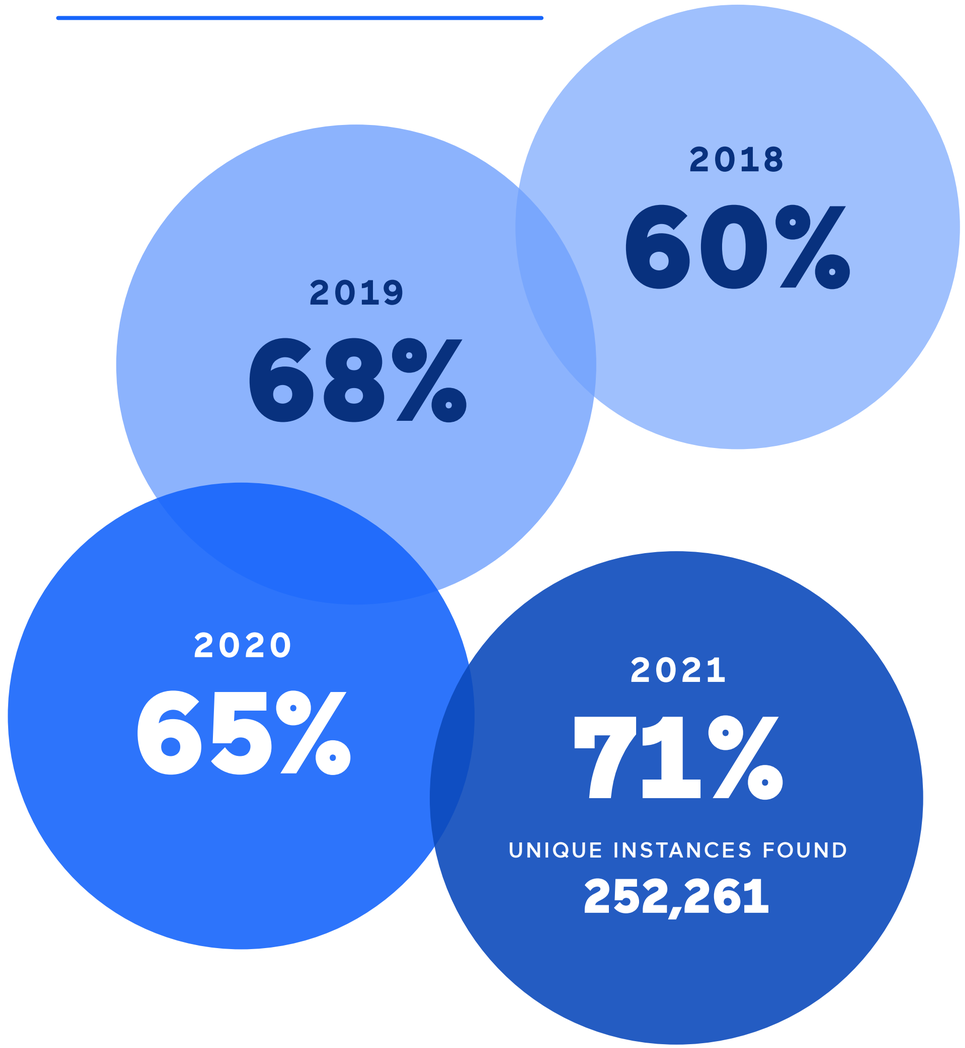
Cross-site scripting is likely the best-known class of web vulnerabilities and many modern frameworks include tools to mitigate it (though not always on by default), so its continued prevalence shows how difficult it can be for developers to close every single gap when processing user inputs.

Learn more about XSS ▸

Cross-site scripting is a big offender, and has made some serious waves in the past. In 2018, British Airways found itself in the media as a cross-site scripting exploit led to a data breach, impacting 380,000 booking transactions. Security researchers suspected that the attack was linked to Magecart, a hacking group that notoriously uses card skimming techniques and malicious code injections.

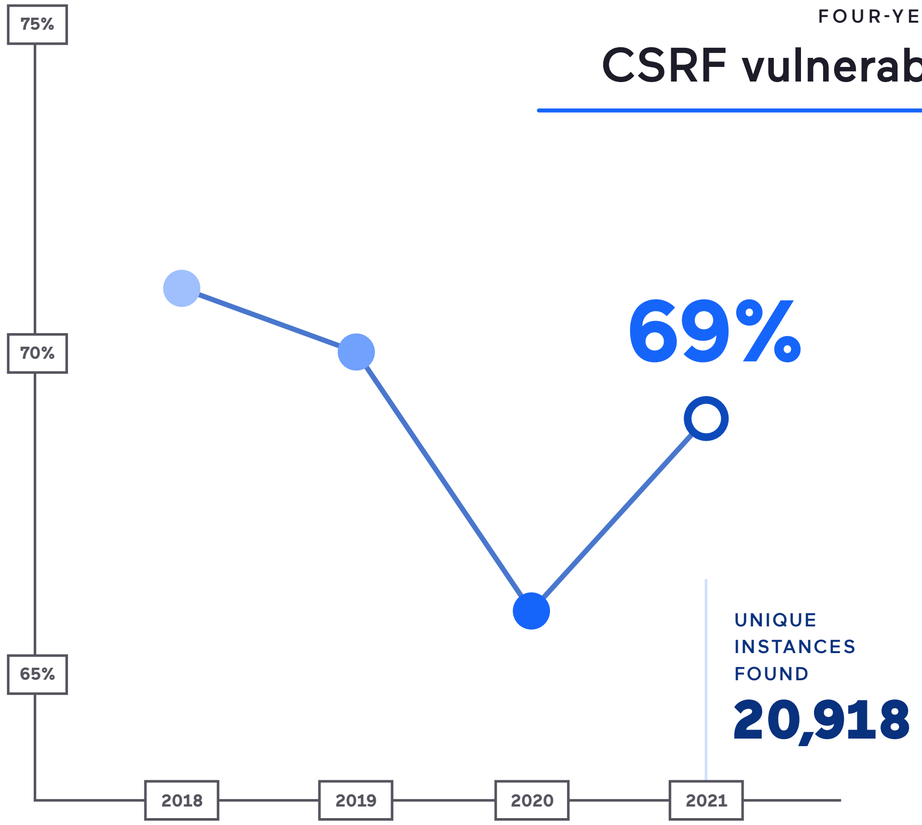
FOUR-YEAR TREND

XSS vulnerabilities



FOUR-YEAR TREND

CSRF vulnerabilities



Cross-site request forgery, or CSRF, is a credentials management flaw that gives attackers the ability to trick end-users into performing unintended operations. Modern web applications routinely retrieve data and resources from external sites, but a CSRF vulnerability allows a threat actor to point the application at a completely different site.

If the user is already authenticated with that site – say, logged into their bank account – malicious requests sent by the attacker via CSRF will be treated like legitimate operations performed by the user on that site. In this scenario, clicking a legitimate-looking link in a spear-phishing email could be enough to authorize a fraudulent multi-million bank transfer.

CSRF attacks often target login forms and, as such, can lead to account takeover (via password reset functionality) or unauthorized access (via newly created malicious accounts). CSRF falls under broken access control, which is the #1 category for AppSec risks in the OWASP Top 10 2021.

Pro tip

Cross-site request forgery is one of the oldest web vulnerability classes and continues to appear frequently despite well-known server-side mitigations, although we can expect advances in browser security to slowly make it less prevalent in the coming years.

Learn more about CSRF ▸

Most often conducted through a social engineering attack via email or an altered link, instances of CSRF will typically only cause harm to the single user that is being targeted. However, if the attacker targets an executive or the primary administrator, that single user's action may also escalate to more dire consequences; for example, granting the attacker access to the administrative interface of the web application and potentially the entire database.

04

LOOKING AHEAD

How do we get a handle on these all-too-persistent flaws and improve security **without sacrificing innovation?**

By embracing a security mindset at every level of the organization and **building the culture, processes, and accountability to support it.**

We're left scratching our heads and wondering why some of these risks are going up in frequency when they should be going away for good. What it comes down to is a disconnect between the reality of risk and the strategic mandate for innovation. It isn't always easy to get everyone on board with security, especially when it seems like security is holding you back from project completion or will be too costly to set up. But what most budget-holders don't realize is that security and innovation go hand-in-hand.

Nobody wants XSS lurking inside of every app – unless they're a bad guy. Talk with your employees about what they need and then roll out a plan to tackle those time-consuming and avoidable risk points. You should always expect your developers to further their knowledge of common vulnerabilities and how to prevent them, especially when it comes to severe flaws we keep seeing year after year.

If you want to convince stakeholders that they need to invest in modern tools, help them **understand that security and innovation need not be in competition with one another** and that both development and security teams are suffering similar stressors. Both are stretched thin as they contend with antiquated tooling, disjointed processes, and too much manual work. This dynamic causes friction and the sense of a zero-sum game between innovation and security.

Although the configuration of an AppSec program varies from organization to organization, every company with a healthy security posture prioritizes **five foundational elements.**

1

Prioritize "secure design" models that cover all the bases, baking security into the pre-code processes behind application architecture.

2

Break down silos between security and development to empower teams to work cross-functionally on security roadblocks.

3

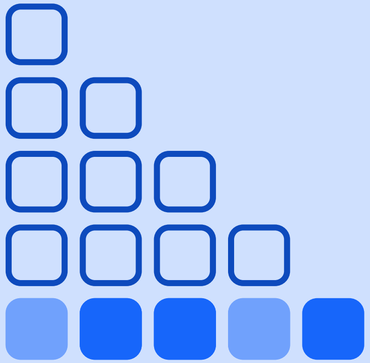
Make security a part of every stage of the software development lifecycle – **scanning both production apps and those in development.**

4

Invest in modern tools that automate everything possible, fold seamlessly into existing workflows, and provide analytics and reporting so you can monitor success.

5

Insist on accuracy from tools that have low false positive rates and provide clear, actionable guidance for developers.



Once more for the people in the back

Security is everyone's job now. As the pace of innovation continues to accelerate, organizations must gain command of their security posture, and the only way to do that is to ensure that security is in the DNA of an organization's culture, processes, tooling, and people.

There's no sugar-coating it. Failures in secure design plus a lack of comprehensive scanning coverage in dev and production continue to expose organizations to potentially catastrophic risk. Especially when you factor in the growing skills gap in cybersecurity, which can mean developers make prioritization tradeoffs without all of the necessary knowledge to minimize risk. Baking the concept of secure design into the software development process makes it a core element of innovation. Developers and security must be held accountable for secure design, comprehensive scanning, rapid remediation, and deep collaboration. A mature DevSecOps approach ensures this will happen.

Initiatives like security champions programs – which aim to improve know-how and spark passion for security – can help you keep talented workers in their seats while also leveling up their skill sets. And with 70% of Information Systems Security Association (ISSA) members citing the cybersecurity skills gap as having an impact on their company, that's a big deal moving forward.[2]

Finding the right cybersecurity talent to fill these critical roles is ever-important because the frequency and breadth of breaches continue to increase, costing more time and money while harming businesses and individuals alike.

The average cost of a single data breach hit a 17-year high in 2021 alone at just over $4 million,[3] with the number of reported compromises reaching 1,862 and impacting 293,927,708 victims.[4] Getting in front of these costly and damaging incidents is a marathon, and the race has already started, so it's time to catch up.

[2] ESG Research Report: The Life and Times of Cybersecurity Professionals 2020

[3] IBM: Cost of a Data Breach Report 2021

[4] 2022 ITRC Annual Data Breach Report

2021

$4 MILLION

AVERAGE COST OF A SINGLE DATA BREACH

1,862

REPORTED COMPROMISES

293,927,708

VICTIMS

So what's next? Frontline defenders who catch, uncover, and mitigate these vulnerabilities every day must stay vigilant and communicate openly with their teams about what they need. And the leaders above those frontline defenders who make decisions about budgets and tooling need to take a serious step back to evaluate what needs to be done. When security is everyone's job and ingrained in an organization's culture, risk reduction doesn't compete with agile frameworks or modern development practices focused on removing barriers and speeding up delivery times – it becomes inherent to them.

As worrisome flaws persist relentlessly, so too can the developers and security professionals striving to build the next innovative applications that keep us connected to the digital world. With an AppSec program that's DevSecOps-focused, developer-friendly, and includes accurate, updated tooling that automates at every step of the way, there is light on the horizon for organizations struggling to reduce risk.

There is light on the horizon for organizations struggling to reduce risk.

Invicti

Invicti Security is transforming the way web applications are secured. An AppSec leader for more than 15 years, Invicti enables organizations in every industry to continuously scan and secure all of their web applications and APIs at the speed of innovation. Through industry-leading Asset Discovery, Dynamic Application Security Testing (DAST), Interactive Application Security Testing (IAST), and Software Composition Analysis (SCA), Invicti provides a comprehensive view of an organization's entire web application portfolio and scales to cover thousands, or tens of thousands of applications. Invicti's proprietary Proof-Based Scanning technology is the first to deliver automatic verification of vulnerabilities and proof of exploit with 99.98% accuracy, returning time to development teams for critical projects and innovation. Invicti is headquartered in Austin, Texas, and serves more than 3,500 organizations all over the world.

www.invicti.com