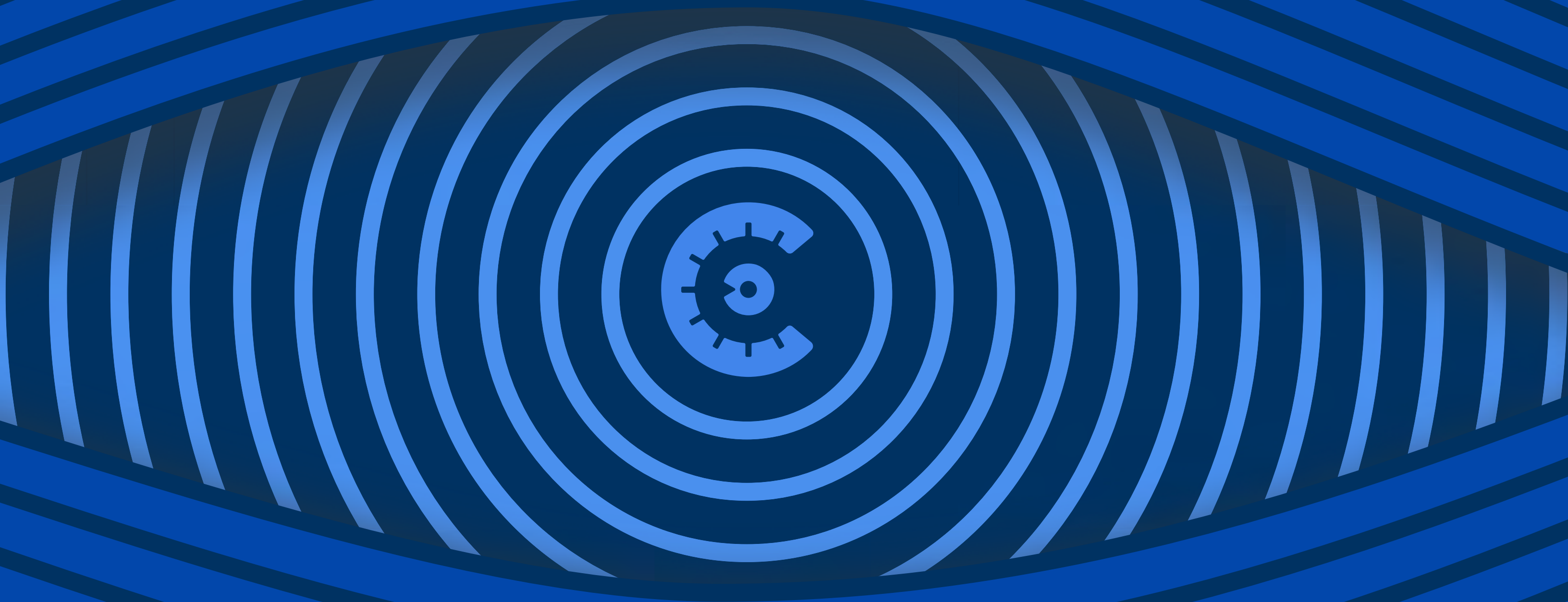


The State Of Pentesting 2021

Table Of Contents

Cobalt

Executive Summary

“There’s no silver bullet solution with cybersecurity, a layered defense is the only viable defense.” - James Scott, Senior Fellow and co-founder of ICIT

As security teams continue to search for ways to deter cyberattacks, it becomes more obvious that multiple lines of defense are the most sound strategy they can take. As business priorities increasingly revolve around engineering output and data privacy, multiple testing types integrate more closely with different stages of the Software Development Lifecycle and link up to discover vulnerabilities before it’s too late.

And yet there are issues that routinely slip past teams’ watchful eyes. As a pentest provider, Cobalt witnesses firsthand how vulnerabilities ranging from Low to High severity make it to software and systems that handle terabytes of sensitive information.

We publish our annual State of Pentesting report to shed light on what those vulnerabilities are, and identify the trends and hazards that impact the cybersecurity community. We collect the data via our proprietary Pentest as a Service (PtaaS) platform, which connects security teams with a carefully curated and thoroughly vetted community of pentesters to examine their systems and software.

This year, we looked at data from 1602 pentests performed in 2020 to learn about the assets getting tested, the vulnerabilities being discovered, and how that data changes across different industries and company sizes. We identified an interesting trend: our customers have been struggling with the same top 5 vulnerabilities for 4 years in a row. Despite the fact that they are well-known to the industry, teams struggle to effectively remove and prevent issues like Server Security Misconfigurations and Cross-Site Scripting from their environments.

To understand why that was, we surveyed 601 companies (not Cobalt customers) to learn how they pursue secure development, how they pentest and remediate vulnerabilities, and where there is room for process improvements for both internal teams and security vendors.

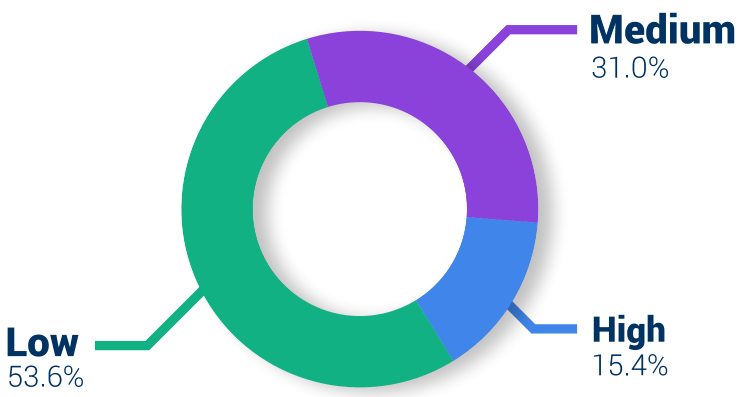
What we found was an interesting mixture of pain points, workflow challenges, and suggestions on how pentesting can evolve as a layer of defense that can help validate the effectiveness of all the other controls that come before it.

Vulnerabilities

We began the hunt for trends by examining the data from our Pentest as a Service (PtaaS) platform. We looked at the types of tests we performed and the types of security issues our pentesters found in 2020. From our sample of 1602 pentests, the majority of our customers came to us specifically for Web Application testing or Web App and API testing—nearly 75% of them. The rest of our sample was made up of combinations of Mobile, External Network, and Internal Network testing.

How much risk are teams managing?

When we discover vulnerabilities, we rate them according to the OWASP methodology, based on their likelihood of exploitation and impact if exploited. Likelihood includes the level of skill an attacker would need to exploit a vulnerability, the availability of documented exploits, and the relative ease of exploitation. The Impact includes the effect on the confidentiality, integrity, and availability of data or systems, as well as potential financial or reputational losses.



VULNERABILITY RISK RATING

High: Vulnerabilities that introduce the most risk of disruption, due to the high impact and high likelihood. Cobalt recommends addressing vulnerabilities at this level as quickly as possible.

Medium: Vulnerabilities that introduce a moderate amount of business risk. These include risks with high impact but low likelihood, and risks with low impact but high likelihood. Cobalt recommends addressing these vulnerabilities to improve the overall security posture.

Low: Vulnerabilities that introduce a relatively small amount of risk, but could still have some impact or likelihood. Cobalt recommends addressing these vulnerabilities when there are no higher priority items.

High-risk vulnerabilities were in the minority among our customers, while more than half of the Findings we discovered were rated Low. However, we still recommend that clients address the Medium- and Low-risk vulnerabilities, because an attacker could find a way to chain several of those less-severe findings together to gain greater access. We explore this observation in more detail under "Remediation."

Why Three Groups? A lot of pentesting organizations group vulnerabilities differently, having a Critical category for the most Likely and most Impactful vulnerabilities, and an Informational category for vulnerabilities with a very low risk. Cobalt sticks with the High, Medium, and Low categories because labeling one item as "Critical" can make other still-risky vulnerabilities sound less important, and labeling an issue as "Informational" can make it sound like it's not worth fixing.

What vulnerability categories are the most common?

To get a glimpse of what flaws slip past security teams' internal checks, we continue our analysis with the top 5 most common vulnerability categories that our pentester community discovered. Unsurprisingly, 2020's list reveals vulnerabilities that are also outlined in the OWASP Top 10:

Top 5 Most Common Vulnerabilities

1. Server Security Misconfigurations: 28.1%
2. Cross-Site Scripting: 15.5%
3. Broken Access Control: 14.7%
4. Sensitive Data Exposure: 8.4%
5. Authentication and Sessions: 8%

We have analyzed our data for the State of Pentesting report every year since 2018, and these vulnerability categories have consistently been our top 5 for each report. The order in which they appear has shifted over time, but Server Security Misconfigurations have been at the top of the list, and by a significant margin, every year running.

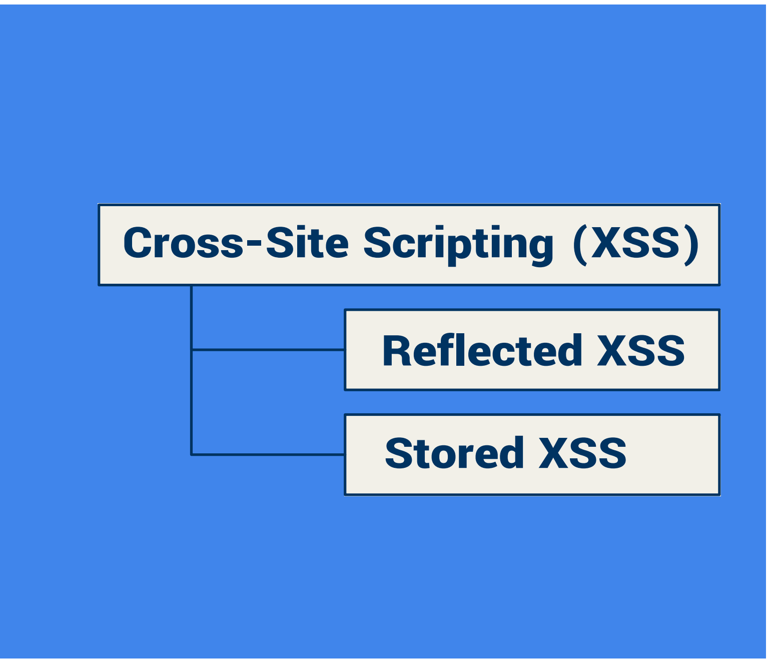
This leads us to believe that security teams are struggling to effectively remove and prevent issues that are well-known to the industry. There can be several reasons for this—gaps in secure development, insufficient investment in security awareness and training, ineffective remediation, or bugs staying open because of low perceived impact and/or lack of resources. We explore more anecdotal data on this point under "Remediation."

Looking deeper into our data

This year we're diving one level deeper into our analysis by observing the specific findings that come up across different methodologies, industries and company sizes. Going this granular helps us determine risk more accurately—for example, with Cross-Site Scripting the risk differs considerably between the Stored and Reflected findings.

But first, a word on taxonomy. Different pentest providers—and cybersecurity organizations as a whole—use different terminology and frameworks to describe their data. To help readers follow our report, here is an overview of the taxonomy we use at Cobalt, which is mostly guided by OWASP best practices:

The building block of our taxonomy are Findings, which are the individual issues our testers discover. Findings are then grouped into a vulnerability category. For example:



Finding #1: An instance of Reflected XSS. When an attacker passes script code to the server in a user-editable location within a request, the application returns it as part of the server's response, which the user's browser interprets and renders as part of the page.

Finding #2: An instance of Stored XSS. When an attacker passes script code to the server in a user-editable location within a request, the application stores it on the server. When another user accesses the affected page, that user's browser interprets and renders the stored code as part of the page.

Vulnerability Category: The root cause for both of these findings is that the application is vulnerable to Cross-Site-Scripting attacks, which happens when the application does not validate and encode user-supplied input properly, so it gets treated as code, rather than as text.

With this system in mind, we looked at the top 5 findings from our entire 2020 database, regardless of vertical or asset type. Here's what we found:

Top 5 Findings For 2020

1. Broken Access Control: Insecure Direct Object References (IDOR): 9.4% of total findings
2. Cross-Site Scripting: Stored: 8.7%
3. Components with Known Vulnerabilities: Outdated Software: 4.1%
4. Broken Access Control: Username/ Email Enumeration: 3.8%
5. Cross-Site Scripting: Reflected: 3.7%

At first glance, the percentages might look small, but that's in part because of the total number of findings we observe in our platform. Among 286 possible options that are grouped into 23 vulnerability categories, these top 5 findings were the most commonly discovered and represent roughly 30% of our data.

It's interesting to note that both of Cross-Site Scripting's variations appeared frequently in our pentests. Despite the fact that this vulnerability became big news over 10 years ago—and some parts of the community have gone so far as to declare it dead—our data and the OWASP Top 10 lists suggest otherwise. And while this type of vulnerability is rarely reflected in big breach stories, its impact on both companies and their customers is not negligible, risking exposure to highly sensitive information.

What are different assets vulnerable to?

While the top 5 overall findings help set the tone for what specific issues security teams can focus on, they won't be helpful to every reader. You'll notice that many of these findings will relate to web assets, and that's partly because the majority of pentests we did in 2020 covered web applications.

To help readers determine the most prevalent threats to their mobile applications, networks, or even combinations of different assets, we've broken the data down to the top 3 findings observed from different testing methodologies.

	1st Place	2nd Place	3rd Place
Web	Cross-Site Scripting (XSS): Stored	Broken Access Control: Insecure Direct Object References (IDOR)	Cross-Site Scripting (XSS): Reflected
API	Cross-Site Scripting (XSS): Stored	Server Security Misconfiguration: Lack of Security Headers	Server Security Misconfiguration: Insecure Cipher Suite
Mobile	Lack of Binary Hardening: Lack of Jailbreak Detection	Broken Access Control: IDOR	Mobile Security Misconfiguration: Absent SSL Certificate Pinning
Internal Network	Components With Known Vulnerabilities: Outdated Software Version	Server-Side Injection: Remote Code Execution	Server Security Misconfiguration: Using Default Credentials
External Network	Components With Known Vulnerabilities: Outdated Software Version	Server Security Misconfiguration: Insecure SSL	Server Security Misconfiguration: Insecure Cipher Suite
Web + API	Broken Access Control: IDOR	Cross-Site Scripting (XSS): Stored	Cross-Site Scripting (XSS): Reflected
Web + Mobile	Lack of Binary Hardening: Lack of Jailbreak Detection	Broken Access Control: IDOR	Insecure Data Storage: Sensitive Application Data Storage Unencrypted
Web + External Network	Cross-Site Scripting (XSS): Stored	Broken Access Control: IDOR	Components With Known Vulnerabilities: Outdated Software Version

None of these flaws are 0-day vulnerabilities. All of them have been well-known to the industry for years, which further strengthens our theory that there is an issue with prevention earlier in the Software Development Lifecycle. We explore this theme in more detail under “Remediation.”

What are different industries vulnerable to?

Some industries are under more pressure than others. For example, as the COVID-19 pandemic disrupted the world in 2020, the Healthcare and E-Learning industries saw an increase in cyber attacks. The FBI reported a 30% increase in attacks against E-Learning targets, and many hospitals and vaccine manufacturers became the targets of phishing campaigns. Based on news stories like these and our pentesting experience, we were interested in seeing what the data said on the industries that we pentest most frequently.

	1st Place	2nd Place	3rd Place
SaaS	Cross-Site Scripting: Stored	Broken Access Control: Insecure Direct Object References (IDOR)	Cross-Site Scripting: Reflected
Healthcare	Cross-Site Scripting: Stored	Broken Access Control: IDOR	Components with Known Vulnerabilities: Outdated Software
Fintech	Broken Access Control: IDOR	Cross-Site Scripting: Stored	Components With Known Vulnerabilities: Outdated Software
Insurance	Cross-Site Scripting: Reflected	Cross-Site Scripting: Stored	Components With Known Vulnerabilities: Outdated Software
E-Learning	Broken Access Control: IDOR	Cross-Site Scripting: Reflected	Cross-Site Scripting: Stored

*Study limitation: the majority of this data relates to Web applications/API assets. For a detailed breakdown of findings per asset type, please refer to "What are different assets vulnerable to?"

While the order varied slightly from industry to industry, it looked like each dealt with similar flaws: variations of Cross-Site Scripting, Insecure Direct Object References, and Outdated Software Version. That, however, doesn't mean that different industries deal with the same level of risk.

Consider the following: Many SaaS and Fintech companies completely rely on their technology to keep the business running. But larger enterprises in industries like Insurance and Healthcare are more likely to have multiple ventures and sources of revenue, so the tested technology could only be a small part of the bigger picture.

What's more, risk doesn't end with the company's business operations. It ultimately translates to impact on end users. For example, weak TLS ciphers in a banking app can be critical for both companies and their customers, while the same problem in a gardening app can be considerably less exploitative.

This is also why we want to highlight one concerning trend: industries which are more likely to handle financial, health, or other sensitive data such as Healthcare, Insurance, and Fintech, are still susceptible to IDOR vulnerabilities. This well-known flaw could allow an attacker to bypass authorization controls and access Personally

Identifiable Information (PII) or Protected Health Information (PHI). These instances are examples of higher risk for both the company and its customers. We recommend that companies whose reports list this vulnerability address it as a high priority.

Aside from considering what industries are vulnerable to, it's also worth exploring metrics that show how vulnerable they are. For example, it would be much less alarming if the number of findings per asset was contained to an average of 1 or 2, than if companies were observing a number closer to 10.

	Avg # of findings per asset	High Risk	Medium Risk	Low Risk
SaaS	6.5	11%	31%	58%
Healthcare	5.4	14%	27%	59%
Fintech	4.9	7%	24%	69%
Insurance	6.2	16%	24%	60%
E-Learning	8.4	17%	38%	45%

In exploring this question, we learned that each industry had room to improve, but E-Learning stood out both with its average number of findings, and what proportion of those carried High and Medium levels of risk.

This could be because many E-Learning businesses had to respond to surging demand for digital solutions during the pandemic. We can easily picture the scenario where businesses were pushed to prioritize faster deployments over secure development. Security teams likely looked to pentests to compensate for what slipped past internal checks—one of many examples of how pentests can act as an effective layer of defense if teams are able to effectively remediate their findings.

Remediation

Earlier in our report we found that the top 5 most common vulnerability categories in our database have stayed the same every year since 2018. When we consult other sources, such as the OWASP Top 10 Vulnerabilities report, this problem goes back even further. For example, Cross-Site Scripting and Broken Access Control have made the list in each iteration since 2003. Security Misconfigurations have also been a prevalent entry, dropping out of the list in 2007 only to reappear in later releases.

This observation raises the inevitable question: why is the industry struggling with the same well-known problems for so long?

To answer this, we wanted to better understand how security practitioners both prevent vulnerabilities and remediate what their pentests discover. We wanted to know if there were any internal issues that limited teams to higher priority items, but also if there were external factors that were coming from their pentest providers.

In Q1 of this year, we spoke to 601 IT security professionals across the United States, Germany, Austria, and Switzerland, all of whom worked for companies with 500 or more employees. They were generally familiar with pentesting, and had previously used pentesting at their companies. More information on our methodology is included in Appendix B.

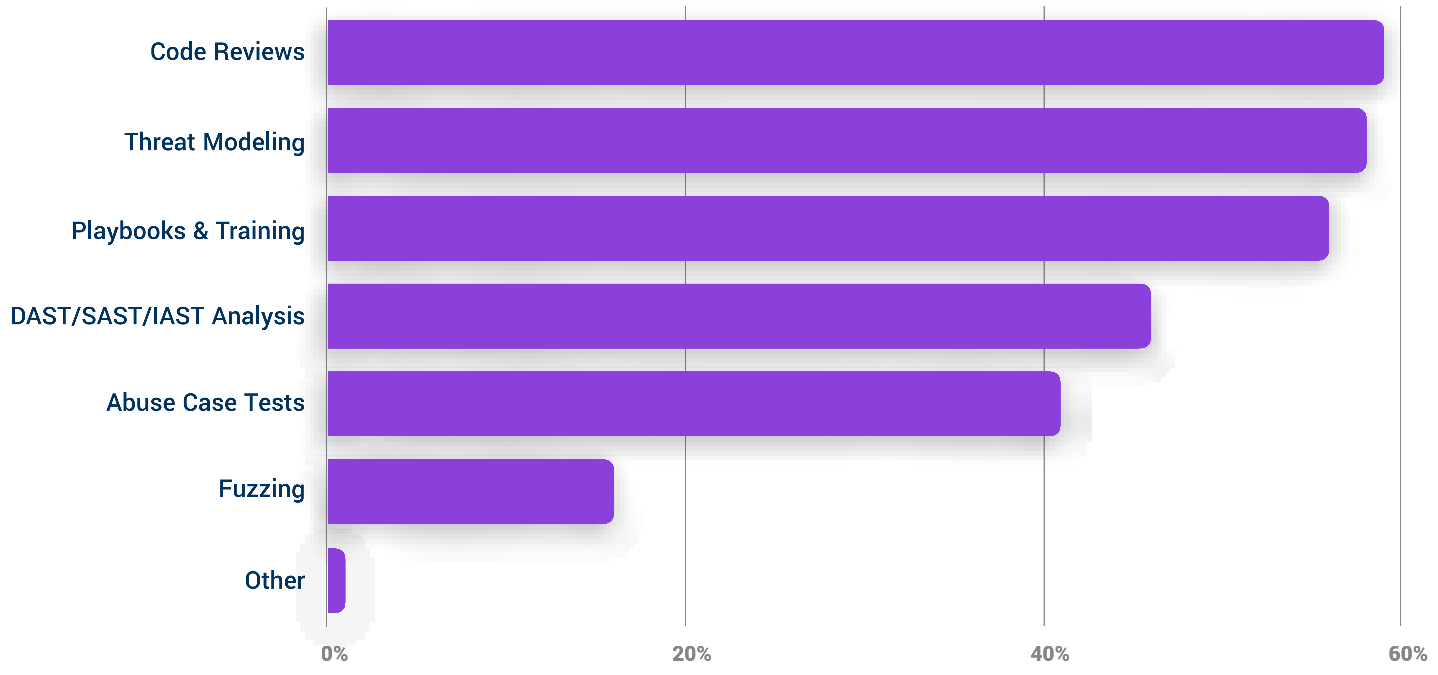
To start, respondents identified and fixed 54 vulnerabilities on average in the last calendar year. And yet **6 out of 10** said they saw these same issues re-emerge at a later date. Why? We have a few theories.

Theory #1: Detecting vulnerabilities before code goes live is still a work in progress.

Before we dive into pentesting-related theories, we want to look at the bigger picture and understand why preventable issues slip past initial checks and only come up after a third party tests for them.

Results: Nearly all respondents (96%) agreed that they have to follow secure development principles to prevent vulnerabilities from re-emerging. We wanted to know how they work towards achieving that:

What security checks does your team have in the software development lifecycle (SDLC) to find and fix vulnerabilities before pushing code to production?



Following secure development principles doesn't always mean that each of these points needs to be at 100%. Of the 43% who don't use code reviews, for example, some might not be using it as diligently as they need to, while others might not need it in the first place. But the fact stands that vulnerabilities slip past these checks, as evidenced by our own data.

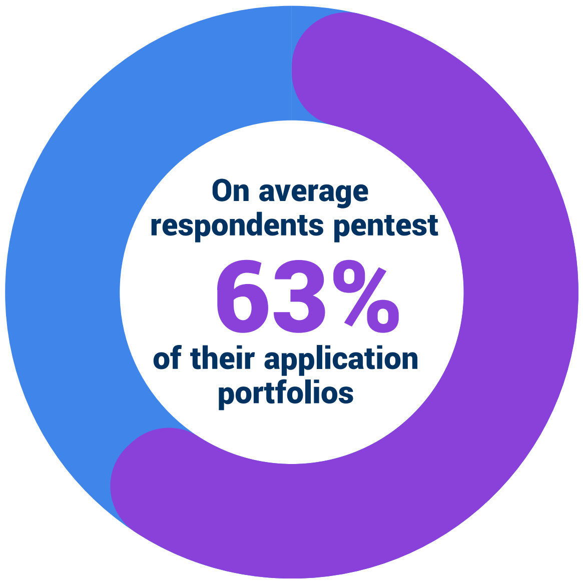
What this graph can tell us is that security teams are using a combination of measures to prevent vulnerabilities early on, which is something we strongly support. It takes multiple approaches to secure one's environment. Pentesting isn't the panacea of application security, but it can validate all the other pieces of a security program and help inform what areas need to be improved.

Theory #2: Teams struggle to detect everything that slips past internal checks because they can't pentest their entire application portfolio.

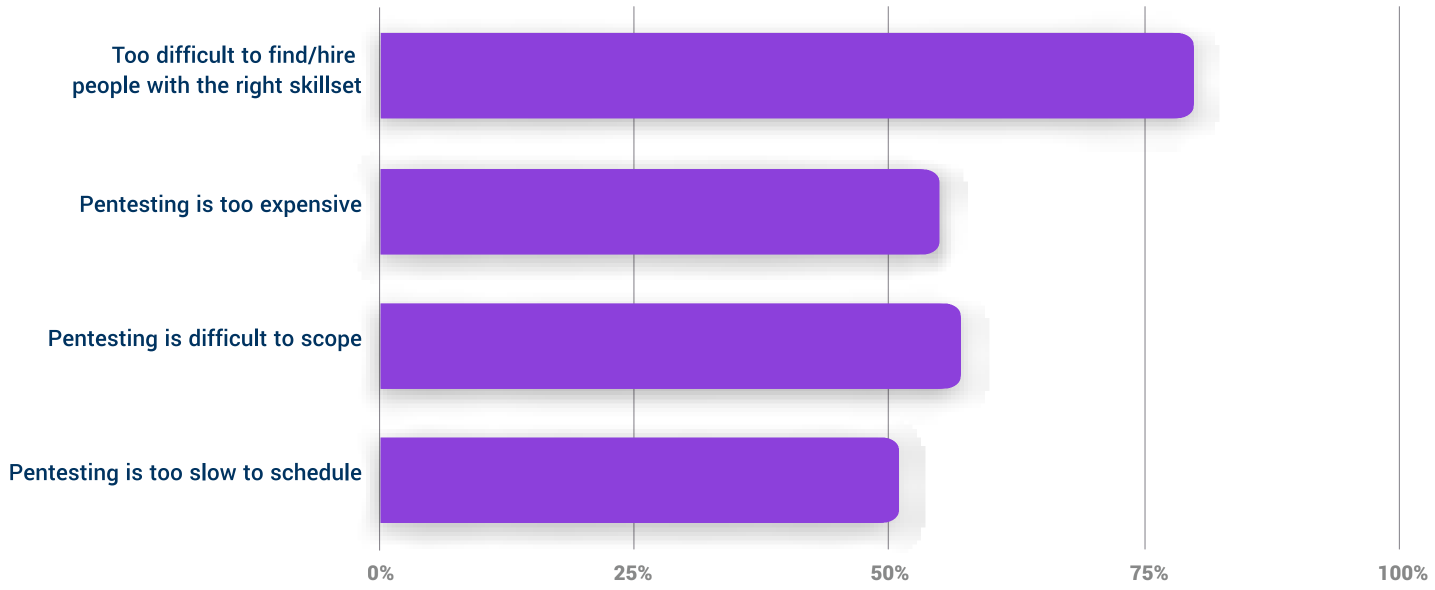
In addition to this theory, we also wanted to see if there were other challenges, for example with procurement and setup.

Results: Even though 78% of study participants agree that pentesting is a high-priority item for their security teams, respondents only conduct pentesting on 63% of their overall application portfolio on average. What challenges could be causing this disconnect?

- **Talent Matching:** 86% of respondents agreed that it is difficult to find and/or hire people with the right skill sets for pentesting.
- **Expense:** 58% of our panelists believe that pentesting is too expensive, with 42% going so far as to say their company does not have the budget to cover it.
- **Scoping Difficulties:** 61% said that pentesting is difficult to scope.
- **Timing:** Over half of the security professionals we interviewed agreed that pentesting is too slow to schedule. How slow, exactly? **Only 22% said they are able to get pentesting scheduled within days**; most have to wait weeks (55%) and some even divulged that it takes their organization months (22%) to get a pentest project off the ground.



Percentage of respondents who agree with the listed statements



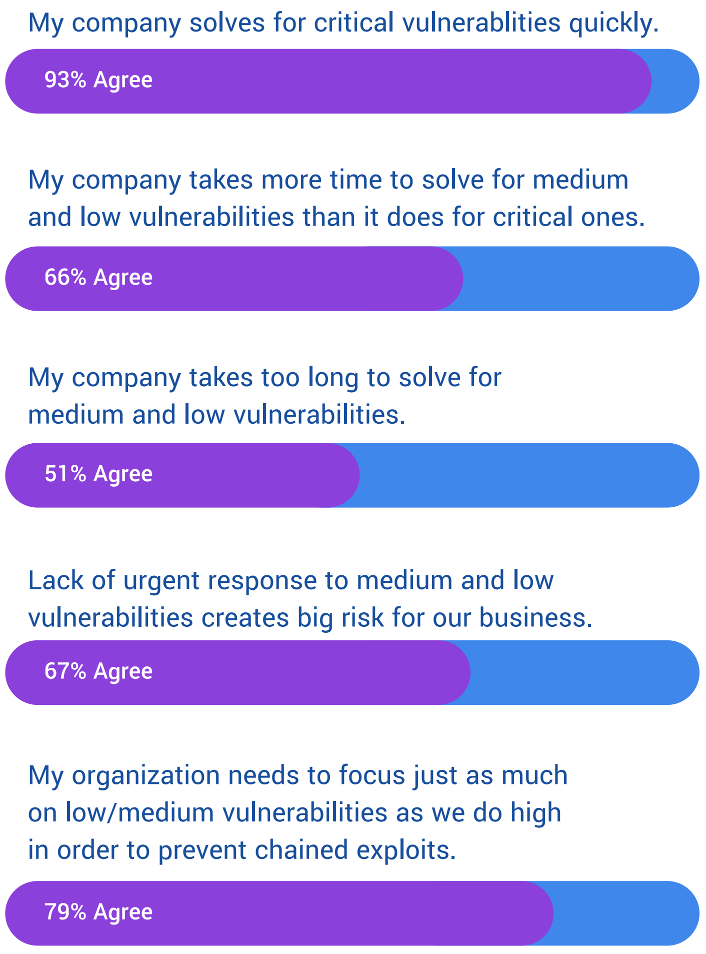
Aside from limiting how much teams can pentest, these pain points also lead to longer periods of time between new code being pushed, and pentests revealing where it is vulnerable. That can translate to months of exposure that no one is aware of and can mitigate.

None of these setbacks are connected to how engineering and security teams work together. Instead, they are symptoms of a wider problem: the established pentesting procurement process makes this security control less accessible, and therefore less reliable. To enable security teams, pentests need to be available with appropriate talent on-demand, with simpler setup and more flexible pricing.

Theory #3: Teams might be taking too long to address Medium- or Low-risk findings.

Results: Nearly all (93%) of respondents reported that their business solves critical vulnerabilities quickly. The response to Low- or Medium-risk vulnerabilities was more sluggish, with two-thirds of our respondents agreeing that it took their team longer to respond to these issues than it did to address High-risk attack points. Over half (51%) confessed that their companies were too slow in responding to these lower-risk issues, and 67% feel that this practice creates significant risk to business, considering that these supposedly low-urgency vulnerabilities can chain together and escalate into very serious problems. To minimize risk and prevent chained exploits, 79% of our study participants agreed that their organizations need to focus just as much on Low or Medium vulnerabilities as they do High.

However, 25% of respondents reported that their company takes up to 60 days—or longer—to address Low/Medium-risk vulnerabilities, and a small but nonetheless surprising segment (1%) of companies don't bother to remediate them at all.

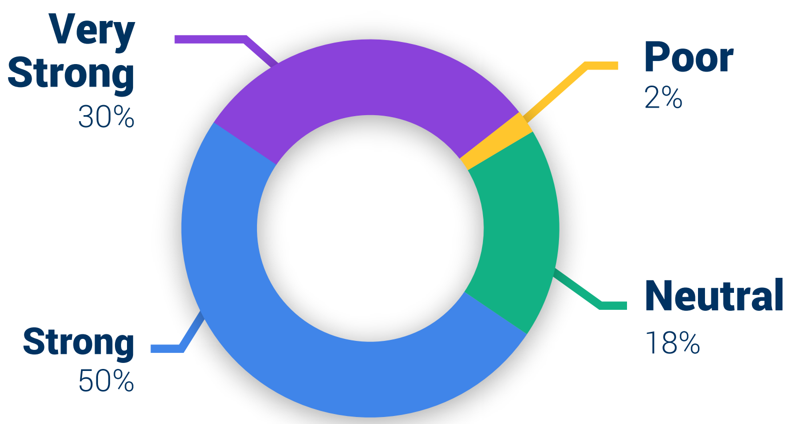


Why is this such poor practice? For a case study on the potentially disastrous consequences of postponing so-called "low-urgency" remediation, look no further than the 2017 Equifax breach. This breach exposed the personal information of 148 million people—more than 40% of the population of the United States at the time—and resulted in a record-shattering $700 million FTC settlement, their CEO stepping down amid massive public controversy, and $1.4 billion spent cleaning up the mess. So how did this catastrophic breach occur? It happened because of a widely-known software vulnerability for which a patch was made available—a patch that Equifax, due to oversight in their internal processes, failed to deploy. Cobalt's Chief Strategy Officer Caroline Wong breaks it down: "This was not a crazy technical problem that lacked a solution. The technical solution was available; this was a lack of people and process innovation."

Theory #4: Remediation teams could be limited to critical findings because of workflow issues.

Results: There is definite room for improvement here. Security and engineering teams still have work to do to effectively collaborate, on remediation priorities and more generally. The outcome is lower-risk findings staying exposed for longer and coming up again at a later test.

How would you describe the current relationship between the security and engineering teams within your organization?

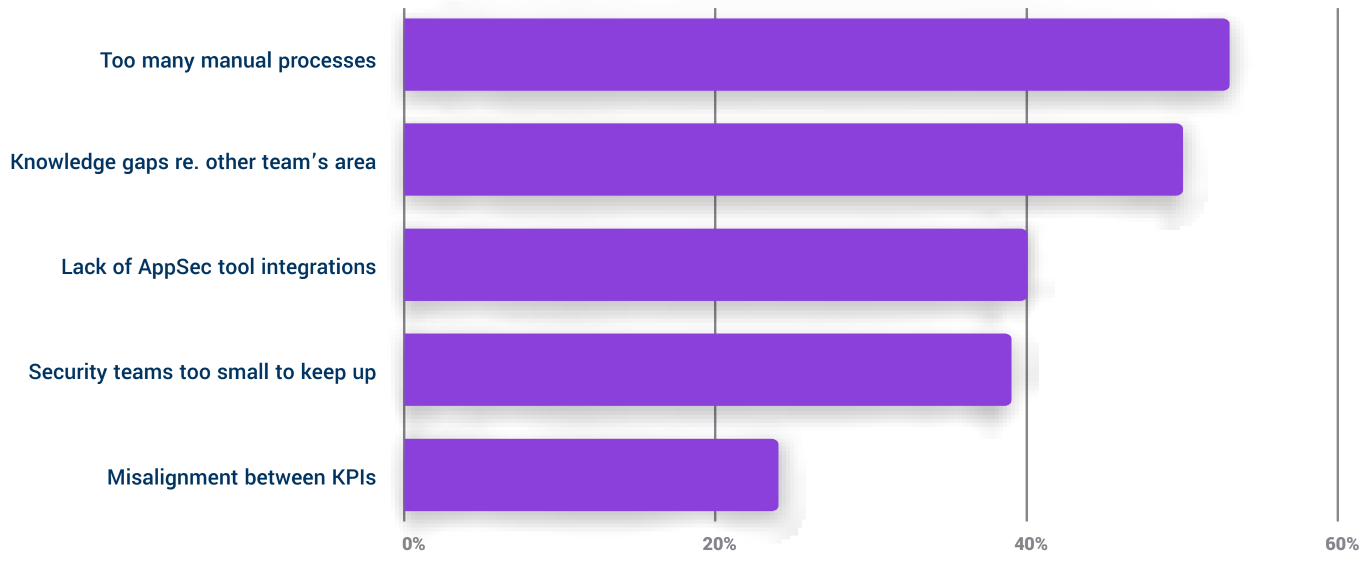


What is holding them back? Largely, the answers are inefficient workflows and lack of automation integrations.

A majority of security respondents agreed that they can collaborate better with their engineering colleagues—much better, in some cases. Half of our respondents reported that the relationship between security and engineering teams within their company was strong, but with room for improvement; 18% said the relationship between these teams was neutral, in that their teams occasionally worked together, but not as often as they should; and 2% of respondents called this relationship poor, reporting that their team rarely worked together with engineers at their company. Only 3 in 10 respondents were able to happily report that their company's security and engineering teams were "intertwined."

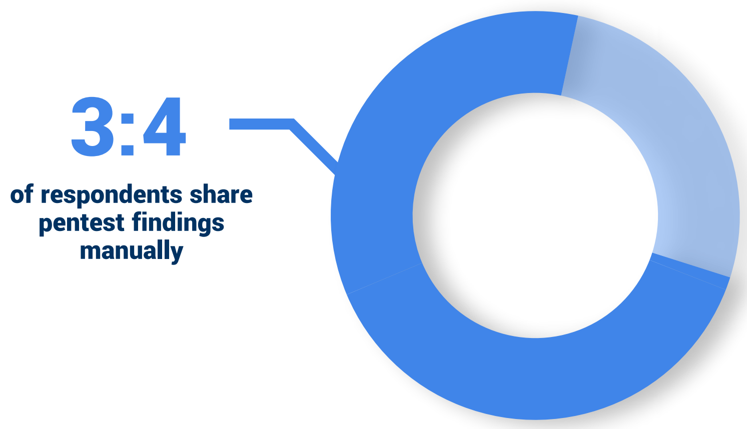
So what are the biggest challenges teams need to address in order to improve DevSecOps workflows? We posed this question to our respondents and here is what they had to say:

What are the biggest challenges your team faces when implementing DevSecOps?



Let's focus on manual processes and AppSec tool integrations, particularly around remediating pentest findings. Once security teams have pentest findings in hand, how are they sending them over to their colleagues in engineering?

Nearly three-quarters of respondents said that this was handled manually within their organization—either the findings were manually pushed to issue tracking software (39%) or worse, the engineering team was manually sent a PDF shared by the pentest provider (34%). Perhaps then it is not surprising that when there was misalignment with remediation priorities, half our respondents said that their engineering teams blamed workflow inefficiencies.



Why is this important?
Most of the time the engineering team also makes up the remediation team—and asking them to address Low- and Medium-risk findings with the same urgency as highly critical ones is a big request when you factor in their commitments to roadmap execution, business enablement, and customer satisfaction. Clearly, security teams cannot pile requests onto these teams unless they explore ways to free up their time with more efficient and automated processes. For example, engineering is not involved in pentesting setup or execution and they rarely have a say in how findings are packaged and shared with them—these are all handled by the security team. Therefore, the responsibility lies with the latter to improve this workflow in order to push toward DevSecOps and remediate more effectively.

Conclusion

Our data strongly suggests that security teams are still struggling with the same well-known vulnerabilities that have plagued the industry for years. Flaws like Cross-Site Scripting and Broken Access Control, which were among the top 5 vulnerabilities in our 2020 data, have also been listed as a prevalent security issue in the OWASP Top 10 every year since 2003.

As much as teams aim for secure development, we conclude there are still gaps in prevention and remediation. As to what might be causing them, and where pentesting plays a part, we discovered the following:

1. The majority of teams pursue some form of secure development, but they don’t catch everything. Pentesting is still a valuable and effective layer of defense.

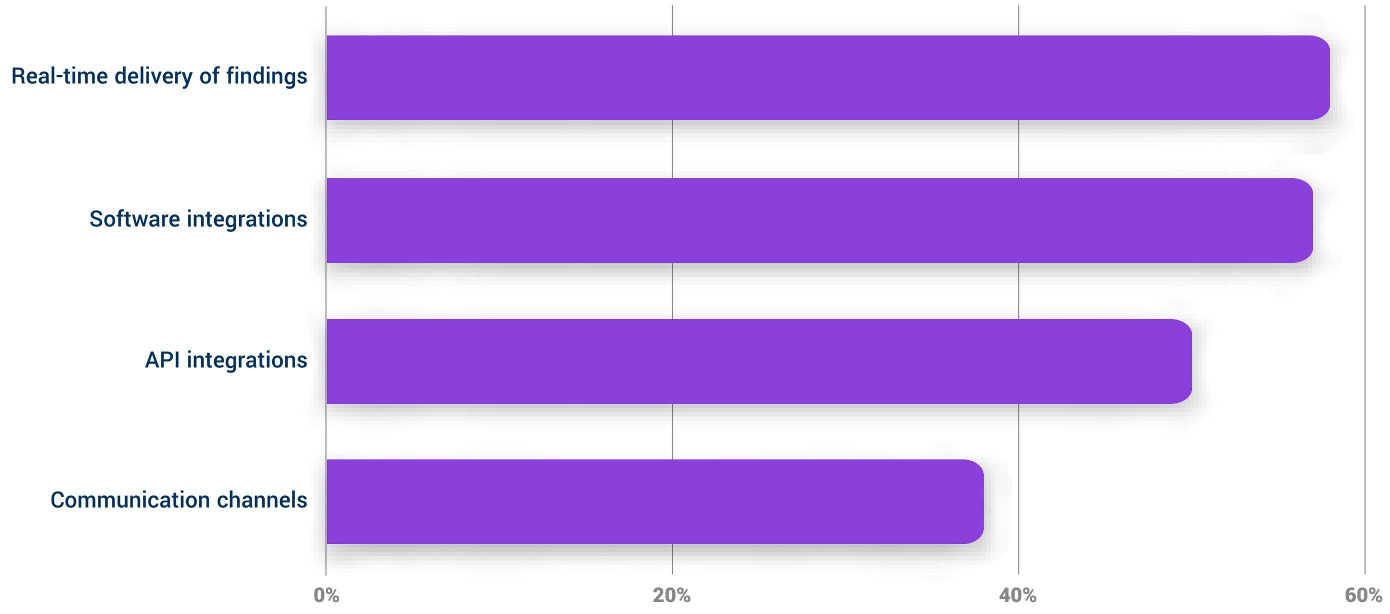
2. To empower security teams to test new code earlier, pentesting needs to be faster and easier to set up, and more financially accessible.

3. Ignoring or delaying remediation for Medium- and Low-risk pentest findings can escalate into a larger issue that is more costly to fix.

4. What holds remediation teams back with fixing pentest findings are too many manual tasks and a lack of tool integrations.

For the final point, we tested a few innovation ideas with our survey respondents to learn what they felt is the best way to improve workflows and get more done with less. The most popular was real-time delivery of findings instead of a PDF report (57%), closely followed by choosing a provider that offers software integrations that automatically share information and status updates between both teams' dashboards (56%). Half our respondents wanted to see API integrations that allow for data consolidation and visualization, and an honorable mention went to dedicated communication channels for security, engineering, and pentesters to collaborate (37%).

What do you think is the best way to align pentesting with both security and engineering roadmaps?



These very innovations and more are what define the movement toward Pentest as a Service (PtaaS). Cobalt's Pentest as a Service (PtaaS) platform, coupled with an exclusive community of testers, delivers the real-time insights you need to remediate risk quickly and innovate securely.

Learn More at [cobalt.io]

The PtaaS Advantage

Traditional Pentesting	Pentest as a Service
Tools	
Disconnected Few to no shared digital tools, findings are collected in a PDF report and sent via email	**Integrated** Cloud platform hosts all relevant pentest information and distributes it via bi-directional integrations with issue trackers, or an open API
Alignment With DevOps	
Restricted Pentests have to be scheduled months in advance and happen once or twice a year despite frequent code deployments	**Flexible** On-demand tests begin in 24 hours for either ad-hoc needs or as part of a continuous program in sync with code releases
Workflows	
Siloed Limited information on bug reproduction and communication overhead between engineers, pentesters and security teams	**Collaborative** Descriptive findings delivered in cloud platform where teams can collaborate in real time
Analytics	
Vague Limited exposure to data on vulnerabilities' criticality and distribution along the codebase	**Detailed** In-platform dashboards or open API provide details on aggregated risk, vulnerability severity and distribution across assets

cobalt.io

Appendix A: How to fix and prevent the most common vulnerabilities

The following chart contains information about the 13 vulnerabilities we mentioned across this report. It includes a short summary of the Finding, and broad advice for ensuring that your assets are protected against them.

We draw these recommendations from the expertise of our security researchers and pentesters, and from established cybersecurity communities' best practices, such as the OWASP's Cheat Sheet Series and the SANS CIS Controls. The guidance in this report is also documented in our platform.

Broken Access Control: Insecure Direct Object References (IDOR)

What is it:

Due to the way a website or application handles resources, one customer's file has an ID in the URL, for example "file123", which corresponds to that file's location in the database.

An attacker who can access "file124" changes the resource ID number to a different number, such as "file123" and can view the other customer's file, or anything else in the database.

How to fix it:

1. Use per-user or per-session indirect object references. For example, instead of using the resource's database key, a drop-down list of six resources authorized for the current user could use the numbers 1 to 6 to indicate which value the user selected. The application has to map the per-user indirect reference back to the actual database key on the server.
2. Check access, and ensure that each use of a direct object reference from an untrusted source includes an access control check to ensure the user is authorized for the requested object.

Stored Cross-Site Scripting (XSS)

What is it:

A web form has a field on it that accepts user text input, then stores that input somewhere on the server's database. For example, a name field on a social media site.

An attacker accesses that web form, and enters a line of code, which the server accepts and stores in the database.

The next time a user pulls up data that includes the attacker's input (such as accessing the attacker's social media profile), their browser attempts to run whatever code they entered.

How to fix it:

1. Treat all user input as untrusted data.
2. Never insert untrusted data except in allowed locations.
3. Always input- or output-encode any data that comes into or out of the application.
4. Create a whitelist of allowed characters.
5. Use a well-known and secure encoding API for input and output encoding, such as the OWASP ESAPI.
6. Never try to write input and output encoders unless absolutely necessary. Research and use an existing output encoder.

Reflected Cross-Site Scripting (XSS)

What is it:

A website passes a piece of information in a URL or other user-editable location, which the server reflects back when that URL is sent.

An attacker creates a URL that contains a string of code in that parameter in the URL. When they reload the page on their own system, or send that URL to another user, the browser accepts that code string and performs whatever actions the string describes.

Note: While Stored and Reflected XSS attacks function differently, the methods for remediating them follow the same design principles.

How to fix it:

1. Treat all user input as untrusted data.
2. Never insert untrusted data except in allowed locations.
3. Always input- or output-encode any data that comes into or out of the application.
4. Create a whitelist of allowed characters.
5. Use a well-known and secure encoding API for input and output encoding, such as the OWASP ESAPI.
6. Never try to write input and output encoders unless absolutely necessary. Research and use an existing output encoder.

Using Components with Known Vulnerabilities: Outdated Software Versions

What is it:

A piece of software that runs a website, server, network, or individual computers does not have the latest security patches installed.

An attacker could search databases of documented vulnerabilities to find information about exploiting one of these outdated pieces of software.

How to fix it:

Keep all software up-to-date, especially if a known vulnerability or weakness associated with an older version exists.

It may also be worth considering a vulnerability scanner, which can report software that has gone out-of-date, and any documented vulnerabilities related to it.

Note: For more information about patches and updates to specific software and hardware in your environment, contact the relevant vendors.

Broken Access Control: Username or Email Enumeration

What is it:

A login portal or forgotten password interface, or similar web feature, prompts users to enter their username or email address. When a user enters a valid response, the server displays a message saying it’s valid (such as “We are sending a password reset email”). When they enter an invalid response, the server displays a different message (like, “user not found”).

An attacker can use different responses to determine what items on a list of potential usernames and email addresses are valid. This could be a list of identified users from past breaches, or guesses based on common first and last names in Census data. The attacker can then use their list of legitimate usernames to attack users.

How to fix it:

Ensure that the application does not reveal existing user names or any data associated with them, whether because of a misconfiguration or a design decision.

Server Security Misconfiguration: Lack of Security Headers

What is it:

When a user accesses a website or application, and the server then serves them the requested information, it sends some information in an HTTP header. A lot of these headers instruct the browser how to interact with the website without rendering in the user's window.

Some of these headers can prevent other types of attacks, such as Clickjacking, XSS, and encryption-related downgrade attacks

How to fix it:

Ensure that all HTTP servers implement the following recommended security headers, at minimum:

- HTTP Strict Transport Security
- X-Frame-Options
- X-Content-Type-Options
- Content-Security-Policy
- X-Permitted-Cross-Domain-Policies
  Cross-Origin-Resource-Policy

For more information on these and other headers that your servers may need, see the OWASP Secure Headers Project.

Weak SSL Configuration: Insecure SSL/TLS

What is it:

When a web application sends sensitive information (passwords, credit card details, Social Security Numbers, or other types of PII, for example) it should send this information encrypted. Meaning that it uses complex ciphers to make that data unreadable to anyone other than the intended recipient, and uses secure Transport Layer Security (TLS) protocols to send that information securely.

However, some of these transport protocols have become outdated, meaning it is possible for an attacker to break into older secure communication channels and decrypt the information. Some configurations also allow an attacker to downgrade a communication from a stronger cipher suite to one that they can crack.

How to fix it:

Use only the most up-to-date TLS protocols. Configure your servers to accept only TLS version 1.2 or higher. Do not accept any Secure Sockets Layer (SSL) versions.

Weak SSL Configuration: Insecure Cipher Suites

What is it:

Related to the Insecure TLS detailed above, an Insecure Cipher Suite refers to the method by which secured data is encrypted. Ciphers are the secret codes used to encrypt information and make it impossible to read without knowing the cipher itself. The goal is to ensure that people other than the intended recipient can't decode the message.

However, some suites of ciphers are old enough that they have been "broken", and can be reliably translated back into the original text.

How to fix it:

Create a shortlist of cipher suites that your servers accept, and use only those.

Select only cipher suites that offer 128-bit encryption.

Lack of Binary Hardening: Lack of Jailbreak Detection

What is it:

When a mobile device has been jailbroken, someone has installed a custom version of the Operating System on the device that gives them higher levels of user permissions than the out-of-the-box version. This is sometimes called "rooting" when it gives the user root access to their device.

How to fix it:

Implement mechanisms to detect whether the application is running on a jailbroken or rooted mobile Operating System. This blocks some of the tools and techniques reverse engineers like to use. Like most other types of defense, jailbreak detection is not very effective by itself, but scattering checks throughout the app's source code can improve the effectiveness of the overall anti-tampering scheme.

Mobile Security Misconfiguration: Absent SSL Certificate Pinning

What is it:

Certificate pinning is the process of associating a backend server with a specific X.509 certificate or public key instead of accepting any certificate signed by a trusted Certificate Authority. After storing, or "pinning", the server certificate or public key, the mobile app will only connect to the known server only. This instructs the application not to trust external Certificate Authorities, which reduces the attack surface.

How to fix it:

Pin and hard-code the certificate into the application.

For more information, see the OWASP cheat sheet on Certificate Pinning: https://owasp.org/www-community/controls/Certificate_and_Public_Key_Pinning

Insecure Data Storage: Sensitive Application Data Storage Unencrypted

What is it:

A vulnerable application stores cleartext sensitive information (such as usernames and passwords, PII, credit card data, or SSNs, depending on the application) within a resource that a user with the correct permissions could access.

Even if the information is encoded and not human-readable, an attacker could use various techniques to determine which encoding is being used, then decode the information.

How to fix it:

Configure the application to store any sensitive information in an encrypted format and resource.

OWASP also recommends performing threat modelling on your application, to determine how it handles the following features:

- URL caching (requests and responses)
- Keyboard input cache
- Copy/Paste buffer cache
- Backgrounded application
- Intermediate data
- Logging
- Stored HTML5 data
- Browser cookie objects
- Third-party analytics data

Server-Side Injection: Remote Code Execution

What is it:

For some server use-cases, a piece of software may require input from a remote user to locate and perform its intended function. For example, user-supplied terms for a database search.

If an attacker supplies a line of code that the server recognizes as a command, a vulnerable server may execute that code and perform an unexpected function.

By executing the command, the server or application could give an attacker a privilege or capability that the attacker would not otherwise have.

Command injection can be an issue with wrapper programs.

How to fix it:

The most effective method of eliminating Code Injection vulnerabilities is to avoid allowing software to evaluate code unless absolutely and explicitly necessary.

However, if there is no way to achieve the same result without code evaluation, ensure that any user input is validated very strongly, placing as many restrictions as possible on that user input.

Server Security Misconfiguration: Using Default Credentials

What is it:

When an application, Operating System, networked device, or other piece of software is installed, it may include pre-configured login credentials for an Administrative panel, or it may create this account with a generated password during set-up. This Admin username and password may be written in some published documentation, or it may be available on the Internet.

If this password is predictable, no one changes it on the first access, an attacker could look up the password and gain unauthorized access to the application.

How to fix it:

1. Never use default credentials as it is trivial for an attacker to gain access by providing known or easy to guess credentials.
2. Always change any kind of default credentials as the first step of setting up any kind of environment.
3. Ensure that all passwords meet or exceed proper password strength requirements. Cobalt recommends at least 12 characters, with complex characters (symbols, capital letters, numbers). Longer passwords are more difficult to crack.
4. If possible, consider disabling external access for systems with Administrative features.

While the fixes listed above should improve the security posture of your application and environments, Cobalt still recommends pentesting on at least an annual basis. This report is not a substitute for a pentest. Our database contains well over 100 vulnerabilities, and we have mentioned just a handful in this report. Regular pentesting can catch additional flaws beyond those we mention here, and can find new ones that may get added as you make changes to your systems. You can also learn about these and other vulnerabilities on the OWASP, SANS CIS Controls, and CVE websites.

Appendix B: Methodology

Cobalt’s State of Pentesting report includes two types of data sets:

- Anonymized pentest data collected via Cobalt’s proprietary Pentest as a Service platform (referred to later as “Cobalt’s Pentest Data”);
- Survey responses on questions related to pentest procurement, setup, delivery and following remediation (referred to later as “Survey Collection”)

Cobalt’s Pentest Data Methodology

The data used in this report represents all pentests executed via Cobalt’s Pentest as a Service platform from January 1st 2020 to December 31st 2020. Cobalt delivers third-party pentest services via the Cobalt platform with the help of a highly curated and collaborative pentest community.

The reported data includes information on the types of assets that were tested and their discovered vulnerabilities, which are broken down to vulnerability categories and associated findings. This accounts for a total of 1602 pentests. The data represents large and small companies, a variety of industries ranging from SaaS to Insurance and Fintech, and 4 geographic regions: EMEA, APAC, North America and South America.

One limitation of this study is the high concentration of pentests commissioned for Web applications and APIs. Combined, these represent roughly 74% of all discovered findings, which is driven entirely by customers’ decisions on what to have pentested. Despite the unequal distribution across asset types, categories like Mobile applications and Internal and External networks still had enough statistically significant data to allow the authors to identify prevalent vulnerabilities for each.

Survey Collection

To explore questions raised in the first half of the report, the authors commissioned an additional survey into how security practitioners procure and manage pentests, and how they remediate discovered vulnerabilities. The survey collected responses from 601 security professionals who work for companies with 500+ employees and live and work either in the United States, or in the DACH region (Germany, Austria and Switzerland). To qualify, respondents had to have a general familiarity with and history of using pentesting throughout their careers.

References

- "OWASP Top Ten Web Application Security Risks", OWASP: https://owasp.org/www-project-top-ten/
- "Pfizer/BioNTech vaccine docs hacked from European Medicines Agency", BBC: https://www.bbc.com/news/technology-55249353 (9 December, 2020)
- "Hospitals Suffer New Wave of Hacking Attempts", Wall Street Journal: https://www.wsj.com/articles/hospitals-suffer-new-wave-of-hacking-attempts-11612261802 (2 February, 2021)
- "Zoom tackles hackers with new security measures", BBC: https://www.bbc.com/news/technology-52560602 (6 May 2020)
- "FBI warns of cyberattacks to distance learning", ABC News: https://abcnews.go.com/Politics/fbi-warns-cyberattacks-distance-learning/story?id=75038470 (4 January, 2021)
- "Equifax's data breach disaster: Will it change executive attitudes toward security?", CSO Online: https://www.csoonline.com/article/3411139/equifax-s-billion-dollar-data-breach-disaster-will-it-change-executive-attitudes-toward-security.html (24 July 2019)
- "Equifax data breach FAQ: What happened, who was affected, what was the impact?", CSO Online: https://www.csoonline.com/article/3444488/equifax-data-breach-faq-what-happened-who-was-affected-what-was-the-impact.html (12 February 2020)
- "OWASP Secure Headers Project", OWASP: https://owasp.org/www-project-secure-headers/
- "Certificates and Public Key Pinning", OWASP: https://owasp.org/www-community/controls/Certificate_and_Public_Key_Pinning
- "NIST Password Guidelines and Best Practices for 2020", Auth0: https://auth0.com/blog/dont-pass-on-the-new-nist-password-guidelines/
- OWASP Cheat Sheet Series, OWASP: https://cheatsheetseries.owasp.org/
- SANS CIS Controls (version 8): https://www.cisecurity.org/controls/v8/

Authors

Jay Paz, Director of Pentest Operations and Research at Cobalt

Jay has more than 12 years of experience in information security and 19+ years of information technology experience including system analysis, design, and implementation for enterprise level solutions. He has a robust background in developer supervision and training as well as in major programming languages, operating hardware and software, and major infrastructure application development.

Robert Kugler, Manager, PenOps Research at Cobalt

Robert Kugler is an information security researcher and pentester who has made his passion for breaking things his job. His background spans over 10 years of data protection, security management and research, as well as penetration testing. Robert has helped strengthen the security of companies such as Mozilla, Axel Springer, PayPal, Spotify, Sophos, Sony, Fitbit, and Deutsche Telekom. In the past, he has given several presentations on IoT security, digital self-defense, the security risks of anti-virus software, and discovered 0days.

Contributors

Caroline Wong, Chief Strategy Officer at Cobalt

As CSO, Caroline leads the Security, Community, and People teams at Cobalt. She brings a proven background in communications, cybersecurity, and experience delivering global programs to the role. Caroline's close and practical information security knowledge stems from her broad experience as a Cigital consultant, a Symantec product manager, and day-to-day leadership roles at eBay and Zynga.

Travis McCormack, Senior Pentest Architect at Cobalt

Travis has been working in the security industry for over 5 years and within IT for over 10 years. He began working within network administration and has since applied those skills and knowledge of network communications and troubleshooting to his career in network security and pentesting.

Danilo Simoes Brambila, Staff Data Engineer at Cobalt

Danilo is passionate about Data & Analytics and over the last 5 years has helped businesses to create modern data platforms and extract value from data. Prior to that he had an extensive academic career, where he specialized in the development of scientific software for simulating complex quantum phenomena and also Advanced analytics for interpreting a wide range of experiments in fields such as nonlinear fibre optics, Attosecond light pulse generation and Material Science.