State of Attacks

Annual Report 2021

fluidattacks.com

fluid attacks
we hack your software

Table of Contents

Introduction

This report shows Fluid Attacks' hacking results over a year. It summarizes our findings and analyses from working closely with our customers to improve their cybersecurity. We present the outcomes of proactively attacking their systems, identifying vulnerabilities and prioritizing the necessary fixes to better support business operations. Report of Attacks 2021 will help you to get ideas about which practices you can implement within your organization to protect it against attacks based on common vulnerabilities across a wide range of industries.

It also provides insights that will help software development, technology infrastructure, DevOps, DevSecOps, cybersecurity, auditing and quality assurance teams within organizations to become more aware of regular practices that put companies at risk. Development teams can identify what are the most common mistakes when building digital products. Security and auditing teams can get clues of where to look when assessing and mitigating risks. In addition, QA teams will also have tools that help them prioritize what needs to be fixed in order to decrease impact within their organizations.

Executive Summary

In this annual report, we continue to showcase our work and results to the world. Our mission to detect accurate security vulnerabilities at the speed of business is clearer than ever before. All metrics demonstrate greater contributions for organizations in managing their cybersecurity operations.

Wider adoption of continuous hacking, more vulnerabilities reported and a higher remediation rate across systems summarize what we have achieved in the past year.

Nevertheless, organizations still have room to improve their cybersecurity management, namely in time to fix and security debt.

Businesses can still leverage much more of their resources to generate competitive advantages in the market from cybersecurity management. By scaling some practices, like *breaking the build*, companies can save more time and avoid losing money while doing business. At the same time, this practice provides peace of mind by avoiding the uncertainty of having vulnerabilities in production. Organizations can also make their vulnerability fix rate target more demanding, speeding up time to market and eliminating security debt.

The world goes too fast, and businesses drive this dynamic environment. An agile approach to cybersecurity is more critical than ever.

Key findings

1. Systems under continuous hacking grew 121%.
2. Almost 132,000 vulnerabilities were reported.
3. The median time to fix critical vulnerabilities reached zero days.
4. *Breaking the build* is associated with 1.8X faster vulnerability remediation on average.
5. The proportion of fixes went up by 36% across all reported vulnerabilities.
6. Systems under one-shot hacking: almost no fixes at all.
7. 28% of high and 36% of critical severity vulnerabilities remain open at least three months after detection.
8. Architecture, Credentials, Cryptography, Data, Logs and Services are still the most breached Rules categories across targets.

fluidattacks.com

fluid attacks

we hack your software

CHANGES

Hacked Systems

121% ↑

Continuous Hacking

Systems under continuous hacking grew 121%. Organizations are harnessing the power of high-speed, comprehensive testing to better manage vulnerabilities. With continuous hacking, companies avoid potential losses in their business operations. Data breaches, disruptions, confidentiality violations and more can be addressed more effectively with this approach.

54% ↓

One-shot Hacking

Systems hacked once (one-shot hacking), on the other hand, went down 54%. Fluid Attacks' customers are increasingly interested in catching vulnerabilities quickly, fixing them faster, and keeping pace with the speed of business.

Nearly 20% of these systems, in the period, came from new customers. The drop in one-shot hacking is better explained by the adoption of DevOps and agile development across organizations.

Continuous hacking: service under a subscription model. Target systems are hacked on a monthly basis and during the development cycle of a product. Vulnerabilities are detected at DevOps speed.
One-shot hacking: service that evaluates one version of an application.

Reported Vulnerabilities

32% ↑

Reported vulnerabilities

In the same period, we reported 32% more vulnerabilities. More continuous hacking leads to more reported vulnerabilities, and also more fixes thanks to the capabilities Fluid Attacks provides to its customers.

25% ↑ Mean 124% ↑ Median

Reported vulnerabilities per system

The number of reported vulnerabilities also went up per system. Mean vulnerabilities grew from 479 to 600 (25%); the median reached 154.5, 124% more. This metric is relevant: vulnerabilities did not only grow as a whole but also within each system. Changes to systems are prone to security vulnerabilities, and Fluid Attacks acts as a reliable preventive line in agile environments.

Reported Vulnerabilities

14% ↑

Growth in reported vulnerabilities for Continuous Hacking

The share of reported vulnerabilities from continuous hacking grew by 14%, reaching 94% of all vulnerabilities over the period.

In contrast, only 6% of all vulnerabilities found came from systems under one-shot hacking. Again, this shows the value that continuous hacking has for business. Fluid Attacks provides the most suitable solutions for organizations shifting to DevOps or already managing their IT under DevOps. This is why our customers prefer continuous hacking.

We must remember that a firm benefits from quickly getting rid of vulnerabilities, rather than simply finding a lot of vulnerabilities or, even worse, from finding more severe vulnerabilities when it is too late.

High and Critical Severity Vulnerabilities

All	Continuous Hacking	One-shot Hacking
43.6%	46.5%	36.5%

Systems with high or critical severity vulnerabilities

46.5% of the systems under continuous hacking reported at least one high or critical severity vulnerability. The figure points out that a considerable proportion of software under development miss dangerous holes that could quickly compromise business operations.

Companies must keep performing continuous hacking to address these more potentially damaging vulnerabilities quickly. Computer Weekly has published that the proportion of high and critical vulnerabilities submitted to the NIST during 2020 accounted for 57% of the total. Moreover, there were submitted more CVEs to NIST in the same year than in any previous year.

Source: Low-complexity CVEs a growing concern https://www.computerweekly.com/news/252496201/Low-complexity-CVEs-a-growing-concern
NIST: National Institute of Standards and Technology

Fixes

All

36% ↑

Continuous Hacking

25% ↑

Percentage of fixes

Across all systems, over this period, fixes increased by 36%. When looking only at systems under continuous hacking, remediation went up 25%. This growth was driven almost entirely by continuous hacking. This approach ignites the motivation in organizations to perform remediations.

Companies are embracing a different paradigm in cybersecurity. They are shifting from concentrating on detecting critical vulnerabilities with testing located “to the right” to focusing on remediation rates with testing placed “to the left.” For businesses, there is a far more significant achievement in the latter paradigm.

Time to Fix

67% ↑

Time to fix gap: with and without *breaking the build*

Organizations that *break the build* kept reaping time benefits: for them, the median time to fix vulnerabilities hardly changed in the last year (from 62 to 64 days). In contrast, for organizations that do not *break the build*, this time increased significantly (from 95 to 116 days).

The gap in time to fix between these groups grew by 67%.

Time to fix: the elapsed time between the detection of a vulnerability and its remediation.
Breaking the build: is a security measure applicable to any CI/CD environment to prevent any software author from putting into production a system with open vulnerabilities, in other words, vulnerabilities yet to be remediated.

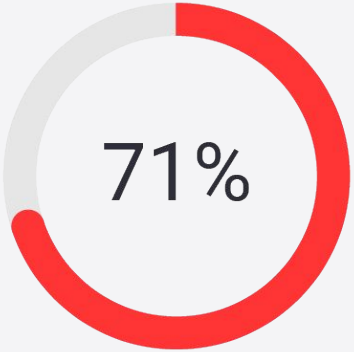
fluidattacks.com

fluid attacks

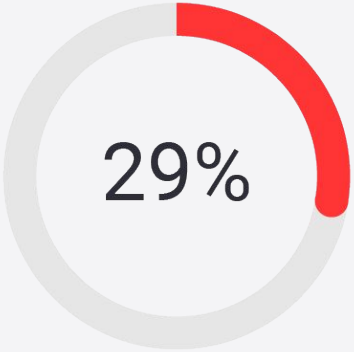
we hack your software

REPORTED FINDINGS

Percentage of Systems in the Period



Continuous Hacking



One-shot Hacking

All Mean 127 days / Median 99 days

Critical severity Mean 11 days / Median 0 days

Average time to fix

Analyzing the remediation behavior of cybersecurity teams, we find two major themes concerning time. First, when it comes to critical severity vulnerabilities, organizations are doing what is expected to fix them almost immediately. A median of zero implies that at least half of these are fixed right away. The second comes from looking at all fixes in the period. We see something close to the opposite: organizations are waiting for too long to fix the remaining vulnerabilities.

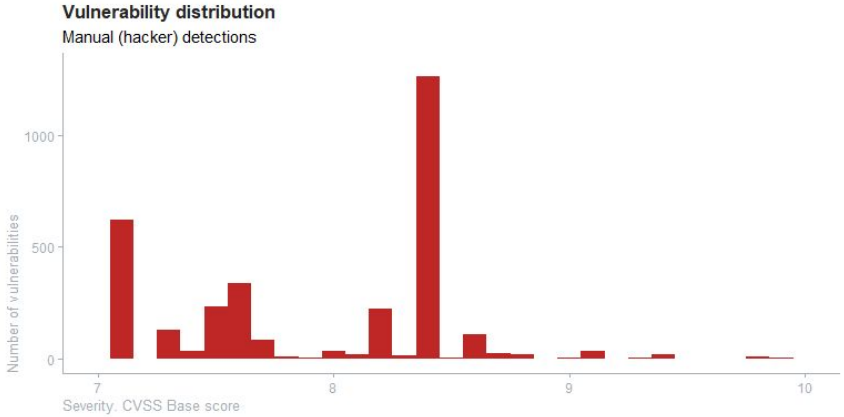
This pattern tells us that companies can do more to streamline the remediation process. One tactic that contributes to this is *breaking the build*, and we will show the positive impact it has on cybersecurity management. Another way to speed up the process is by enforcing remediation more rigorously within the organizations' DevOps operations.

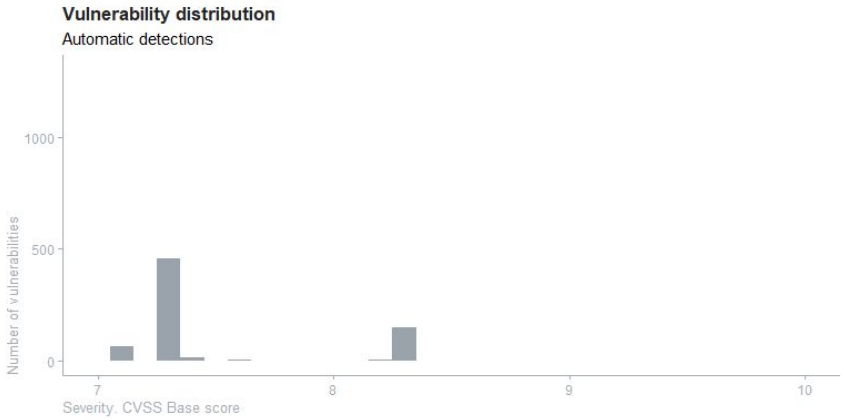
Continuous Hacking

High and critical severity vulnerabilities by source

Moreover, by looking closely at the distribution of vulnerabilities, we observe that ethical hackers play a crucial role in detecting the most severe ones. Our skillful, ethical hackers found roughly 81% of high and critical severity vulnerabilities over the analysis period.

If left to automated tools only, organizations could dangerously miss a large number of vulnerabilities with the highest impact on business operations. These kinds of false negatives have caused multimillion-dollar losses for several companies with worldwide recognition.





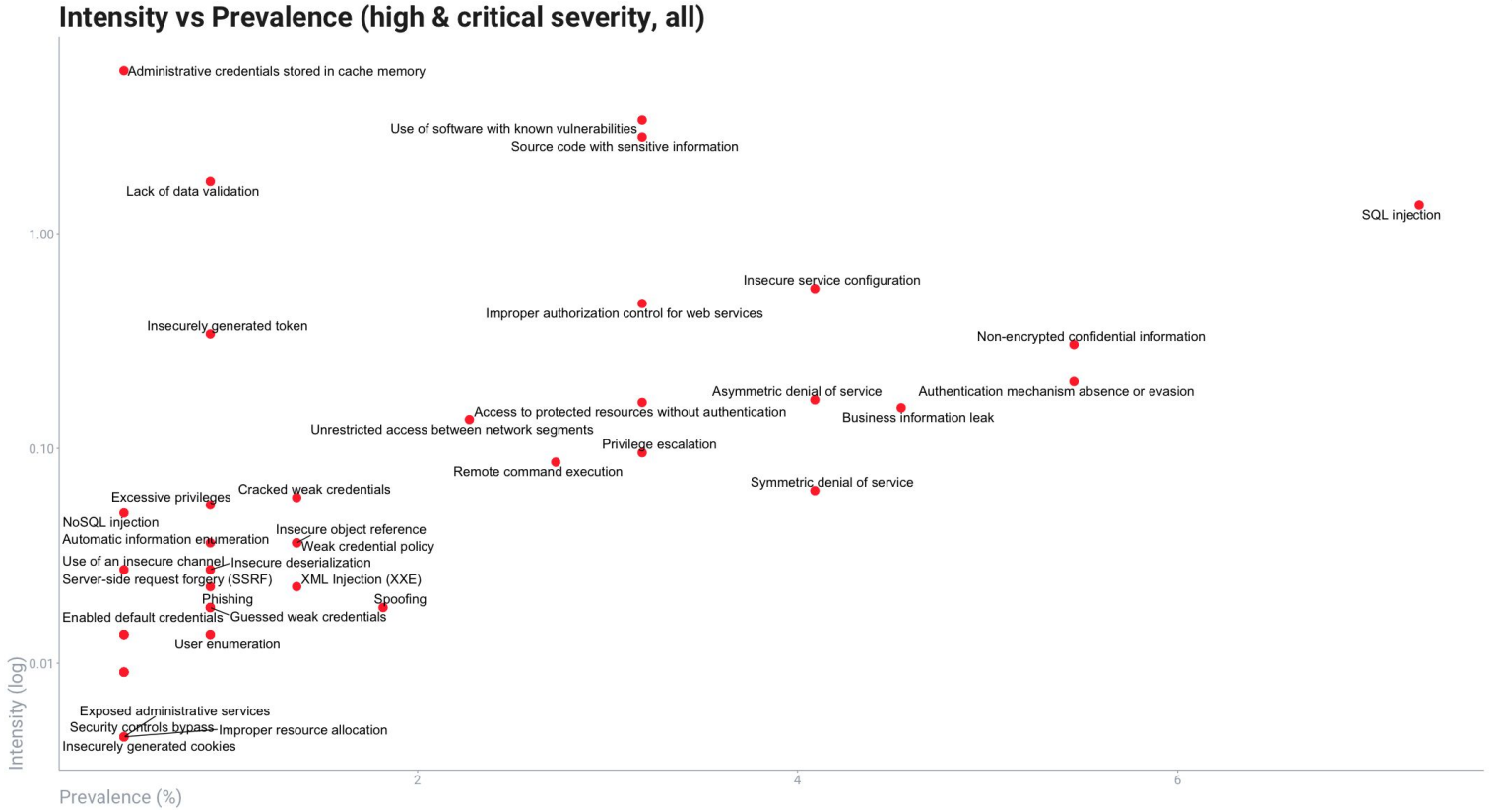
fluidattacks.com

fluid attacks

we hack your software

SEVERITY OF VULNERABILITIES

Severity of Vulnerabilities, All



Intensity: average number of vulnerabilities per system.
Prevalence: share of systems with the finding.

Severity of Vulnerabilities, All

Intensity vs prevalence (high & critical severity)

In the plot, we see the high and critical findings over the period in two dimensions: intensity and prevalence. The first dimension is similar to a density (in log scale) and the second one is a spread metric (in percentage).

We note that well-known vulnerabilities are still present in many systems, such as SQL-Injection, to the top-right, with an intensity of above 1.0 and a prevalence of almost 8%. In addition to this flaw, the following also have an intensity of above 1.0 (i.e., one finding per system on average):

- Use of software with known vulnerabilities
- Lack of data validation
- Source code with sensitive information
- Administrative credentials stored in cache memory

To the right, starting at a prevalence of 4% (vulnerabilities found in 4% or more systems), we find:

- Insecure service configuration
- Symmetric denial of service
- Asymmetric denial of service
- Business information leak
- Non-encrypted confidential information
- Authentication mechanism absence or evasion

fluidattacks.com

fluid attacks

we hack your software

REMEDIATION (FIXES)

Continuous Hacking

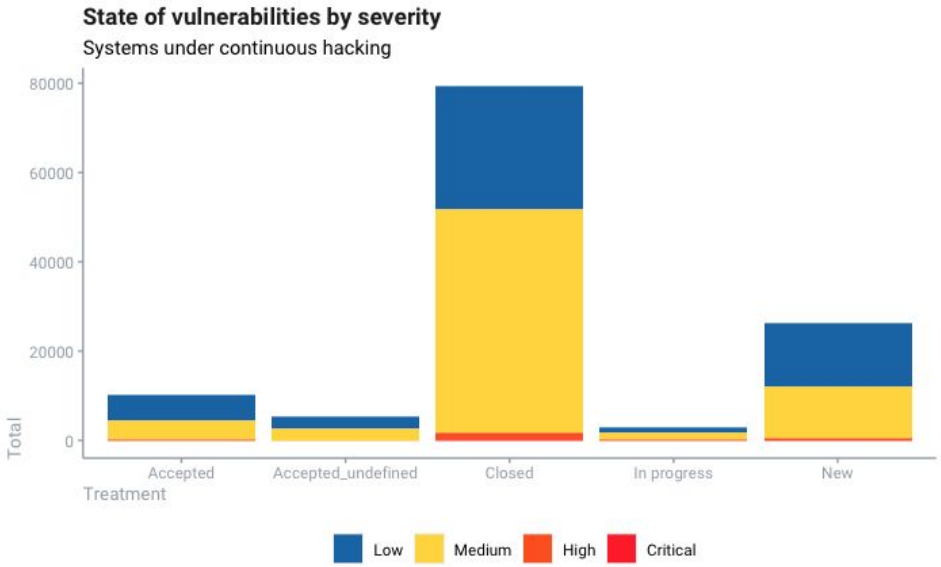
State of vulnerabilities by severity (freq)

The majority of vulnerabilities are fixed (closed). Nevertheless, a considerable proportion of low and medium severity vulnerabilities are accepted.

Furthermore, within high and critical severity vulnerabilities, we see worrying numbers:

- 72% of high are fixed or in progress.
- 64% of critical are fixed or in progress.

This means 28% of high and 36% of critical severity flaws are still in the backlog to be fixed.



Treatment of Vulnerabilities
New: default classification. When reported, all vulnerabilities are classified as New.
In progress: indicates that the customer will fix a vulnerability.
Closed: indicates that a vulnerability has been fixed.
Accepted: indicates that the customer will not fix a vulnerability. However, this classification can be revised in the future.
Accepted_undefined: indicates that the customer will not fix a vulnerability. No revision in the future.

Continuous Hacking

Time to fix (days)

Breaking the build is associated with reducing by more than half the average time to fix; 45% decrease if we take median time to fix. Organizations not enforcing this practice have an enormous opportunity for improvement.

The following are the median times to fix vulnerabilities we found considering continuous hacking and *breaking the build*:

- 64 days (*breaking the build*)
- 116 days (no *breaking the build*)

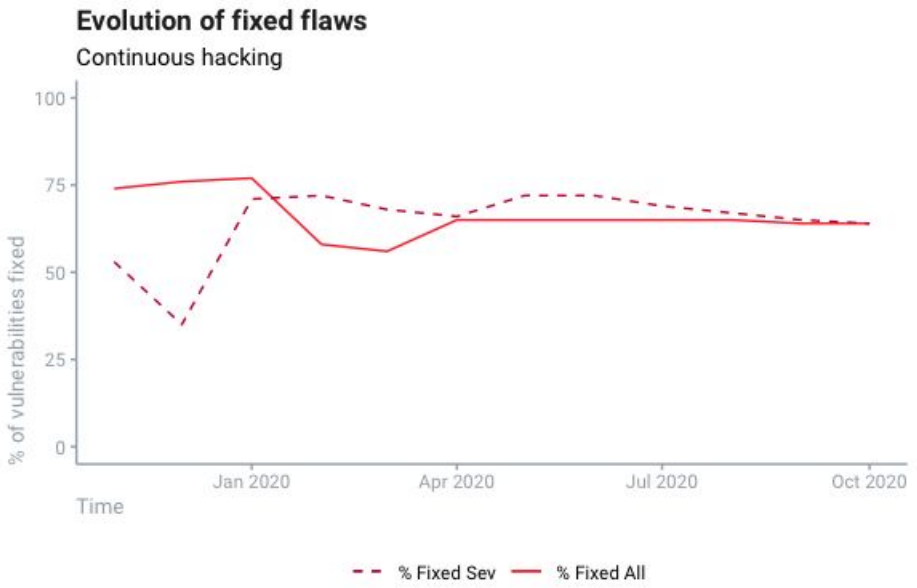
Critical vulnerabilities are fixed almost immediately after detection (median: 0 days; mean: 11 days). On the other hand, there is little variation between the remediation times for high, medium and low severity vulnerabilities, and the figure is not small.

Severity	Average	Median	Vulns
Low	128.48	95	27834
Medium	125.26	100	50025
High	112.67	111	1590
Critical	10.44	0	27

Continuous Hacking

Fixes over time: All vs high & critical

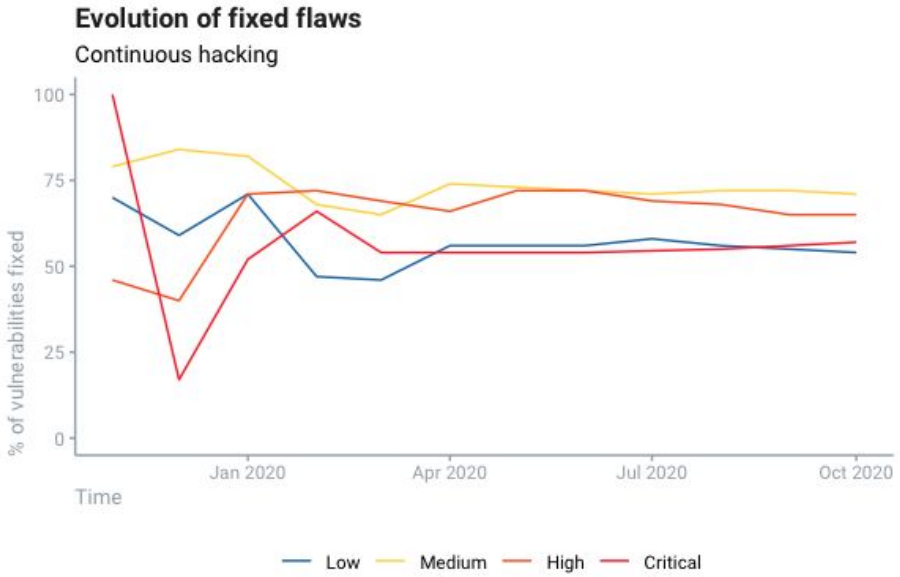
Is there a relationship between fixing vulnerabilities and time? Apparently not. In the plot, we see the rate of fixes over time, considering all vulnerabilities and considering high and critical severity ones. Although there is some variability, the mean fixing rate over the period is 64% for both.



Continuous Hacking

Fixes over time by severity

Now, let's see the same plot, broken down by severity. There are notable differences. First, low and critical severity vulnerabilities end up almost with the same fixing rate (54% and 57%, respectively). Medium severity vulnerabilities end up with a fixing rate of 71%, whereas high severity ones end up with 65%.

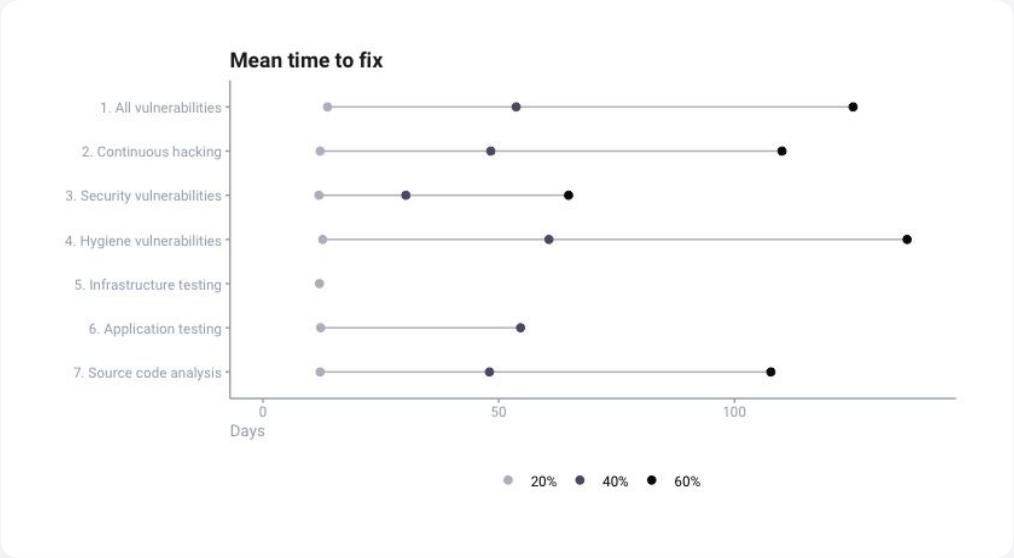


Continuous Hacking

MTTF/MeTTF

Once again, we measured the time taken by our customers to remediate or close vulnerabilities across various conditions and we display their means and medians here. On this occasion, customers were able to reach 60% of fixes for most of the situations.

In the first plot, we begin by comparing all vulnerabilities vs. those from continuous hacking. As noted earlier in this report, the mean time to fix (MTTF) is lower for continuous hacking. This holds true for both metrics, but the MTTF is clearer.



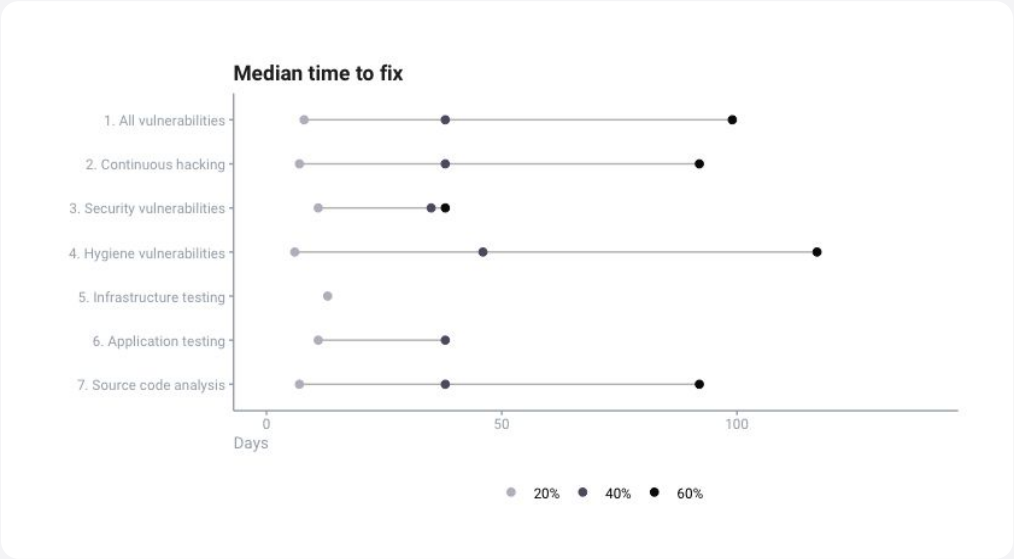
Mean time to fix (MTTF): average time a vulnerability remains open.
Median time to fix (MeTTF): median time a vulnerability remains open.
Here, we use days as the time unit.

Continuous Hacking

MTTF/MeTTF

Security vulnerabilities are fixed significantly faster than hygiene ones (this is more evident in the median time to fix —MeTTF). This suggests that customers seem to use this categorization to decide what to fix.

Finally, two things stand out immediately in relation to the targets' time to fix: infrastructure vulnerabilities are scarcely fixed (less than 40%), and application vulnerabilities are fixed very quickly (there is no difference in time between reaching 40% and 60% fixes). Regarding source code vulnerabilities, we observe a huge delay to fix after 40% is reached.

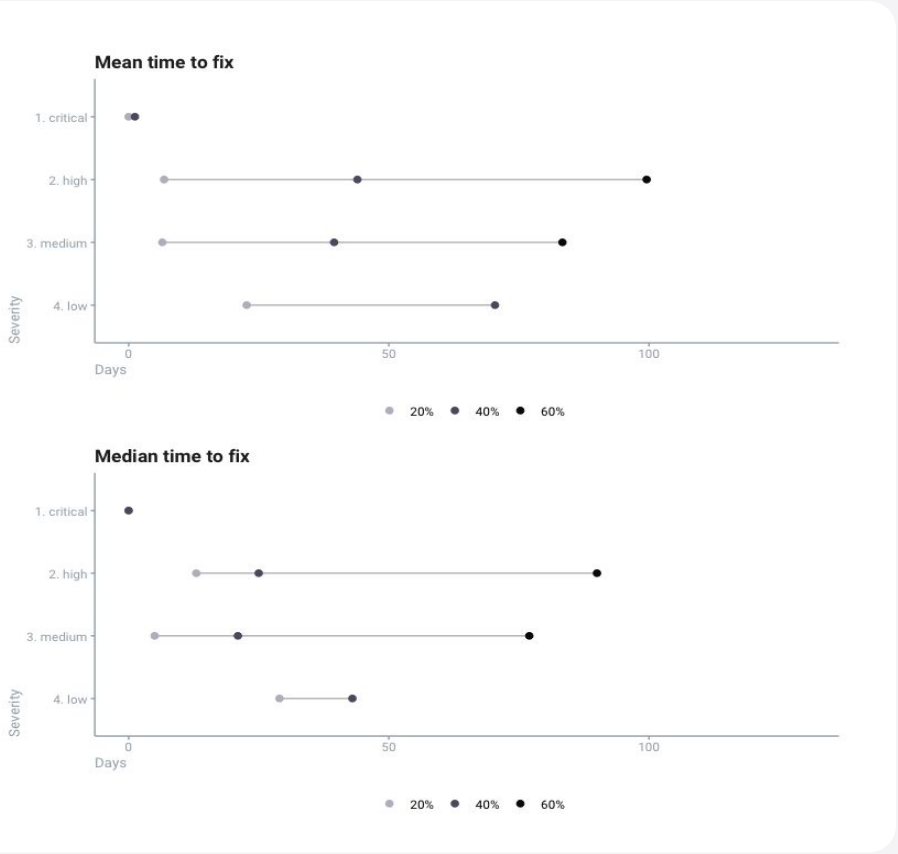


Mean time to fix (MTTF): average time a vulnerability remains open.
Median time to fix (MeTTF): median time a vulnerability remains open.
Here, we use days as the time unit.

MTTF/MeTTF, by severity

Now, let's review how severity is associated with the time to fix. We observe that critical vulnerabilities are fixed almost immediately: the MeTTF is zero days for 40%, and its corresponding MTTF is just 1.2 days.

However, there is a big BUT: high and medium severity vulnerabilities are hardly distinguishable in time to fix. Even worse, high severity vulnerabilities take more time to reach all thresholds compared to medium severity ones. Finally, the plot shows that organizations take longer to start fixing low severity vulnerabilities (20% threshold); they reach 40% remediation after 14 median days but do not achieve 60% as with the other severity groups.

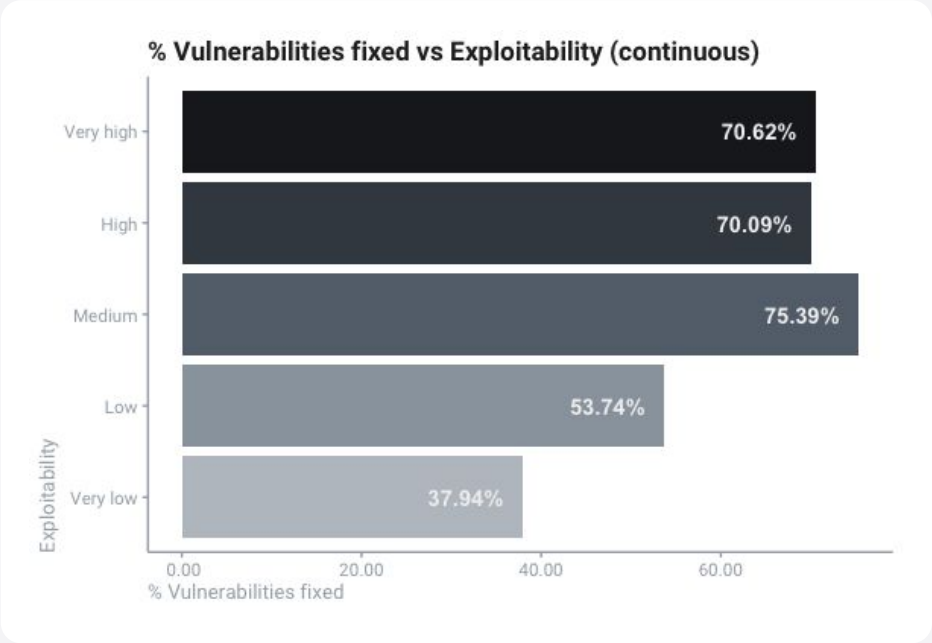


Continuous Hacking

Fixes by exploitability (CVSS)

This chart shows an association between exploitability (as measured by CVSSv3) and the remediation rates. We see a positive correlation suggesting a highly coherent behavior.

Furthermore, very high and medium exploitable vulnerabilities gained more attention in this period. In the previous report, these vulnerabilities had fixing [remediation] rates of 48% and 35%, respectively.



Exploitability (CVSSv3): it is a sub-component of the Base Score in the standard. It measures how easily a vulnerability can be exploited. This metric is calculated based on the following criteria:
Attack Vector (AV) – indicates how the vulnerability can be exploited.
Attack Complexity (AC) – indicates situations that are out of the attackers' control and that are required to exploit a vulnerability.
Privileges Required (PR) — indicates the number of privileges the attacker must have to exploit the vulnerability successfully.
User Interaction (UI) — captures the requirement for a user, other than the attacker, to participate in the successful compromise of the vulnerable component.

fluidattacks.com

fluid attacks

we hack your software

FREQUENCY OF FINDINGS

Hacked Systems

Top 10 findings (by % of systems)

Three out of five common vulnerabilities in the rankings (by systems and persistence) of the previous report appeared again:

- Use of software with known vulnerabilities
- Lack of data validation
- Non-encrypted confidential information

Five new findings made it to the top 10:

- Insecure encryption algorithm
- Use of an insecure channel
- Insecure service configuration
- Source code with sensitive information
- Sensitive data stored in client-side storage

#	Finding	% sys	per	M_s	Me_s
1	F011. Use of software with known vulnerabilities	60%	11352	4.92	5.00
2	F052. Insecure encryption algorithm	56%	1546	2.86	2.60
3	F063. Lack of data validation	53%	2060	5.18	5.30
4	F022. Use of an insecure channel	51%	2284	3.44	3.50
5	F020. Non-encrypted confidential information	48%	5291	3.63	3.50
6	F055. Insecure service configuration	42%	776	4.86	4.60
7	F009. Source code with sensitive information	34%	3123	5.23	5.40
8	F026. User enumeration	33%	218	5.14	5.30
9	F085. Sensitive data stored in client-side storage	32%	769	3.27	3.10
10	F102. Email uniqueness not properly verified	28%	120	4.26	4.30

Persistence: the number of vulnerabilities associated with a single finding.

Hacked Systems

Top 10 findings (by persistence)

One of the vulnerabilities that was left out in the general ranking is **Business information leak**, the most severe in the previous report. However, we found it in a more specific ranking.

#	Finding	% sys	per	M_s	Me_s
1	F011. Use of software with known vulnerabilities	60%	11352	4.92	5.00
2	F020. Non-encrypted confidential information	48%	5291	3.63	3.50
3	F009. Source code with sensitive information	34%	3123	5.23	5.40
4	F022. Use of an insecure channel	51%	2284	3.44	3.50
5	F063. Lack of data validation	53%	2060	5.18	5.30
6	F052. Insecure encryption algorithm	56%	1546	2.86	2.60
7	F019. Administrative credentials stored in cache memory	2%	1268	8.39	8.40
8	F055. Insecure service configuration	43%	776	4.86	4.60
9	F085. Sensitive data stored in client-side storage	32%	769	3.27	3.10
10	F059. Sensitive information stored in logs	20%	700	3.53	3.50

Hacked Systems

Top 5 findings (by persistence) - INF

Most of the top persistent vulnerabilities found across targets were of medium severity. However, within infrastructure targets, there was one high severity finding: **Administrative credentials stored in cache memory**, with a median CVSS base score of 8.4.

Two more vulnerabilities made the ranking with a severity of above 6. First, **Improper authentication for shared folders**. Second, **Business information leak**. As previously mentioned, this vulnerability didn't make the general top 10 ranking, but is one of the most persistent findings in infrastructure targets.

#	Finding	% sys	per	M_s	Me_s
1	F019. Administrative credentials stored in cache memory	1%	1267	8.39	8.40
2	F018. Improper authentication for shared folders	1%	176	6.33	6.30
3	F038. Business information leak	2%	156	6.78	6.80
4	F024. Unrestricted access between network segments	4%	83	5.75	5.30
5	F047. Automatic information enumeration	2%	68	4.25	4.30

Hacked Systems

Top 5 findings (by persistence) - APP

Regarding deployed application targets, **Lack of data validation** is the most persistent vulnerability. This is worrying, given its prevalence in half of the systems we tested. **Insecure encryption algorithm** also made it to second place in this specific ranking.

#	Finding	% sys	per	M_s	Me_s
1	F063. Lack of data validation	50%	1669	5.18	5.30
2	F052. Insecure encryption algorithm	43%	948	2.66	2.60
3	F047. Automatic information enumeration	16%	433	5.61	5.70
4	F011. Use of software with known vulnerabilities	15%	338	5.43	5.00
5	F055. Insecure service configuration	27%	316	5.77	5.40

Hacked Systems

Top 5 findings (by persistence) - CODE

From the source code review, three findings are worth mentioning in this ranking. First, the most persistent one is the **Use of software with known vulnerabilities**. DevOps teams can improve their decisions regarding third-party software known to be flawed.

In the second and third positions, we have two widespread vulnerabilities linked to sensitive information: on the one hand, the **absence of encryption** for this information; on the other hand, the presence of this information in **the source code**. Likewise, organizations can and should improve the handling of confidential information.

#	Finding	% sys	per	M_s	Me_s
1	F011. Use of software with known vulnerabilities	52%	11001	4.91	5.00
2	F020. Non-encrypted confidential information	41%	5203	3.58	3.50
3	F009. Source code with sensitive information	33%	3120	5.23	5.40
4	F022. Use of an insecure channel	44%	2211	3.42	3.50
5	F059. Sensitive information stored in logs	18%	695	3.52	3.50

fluidattacks.com

fluid attacks

we hack your software

NON-COMPLIANCE WITH RULES

Non-Compliance With Rules

Common violated rules across top 20:

- **Data.R176.** *Restrict system objects*
- **Architecture.R266.** *Disable insecure functionalities*
- **Logs.R077.** *Avoid disclosing technical information*
- **Services.R262.** *Verify third-party components*

Rules categories without problems according to the testing targets, worth mentioning:

- INF: authorization
- APP: certificates
- CODE: authentication, authorization and credentials

Cryptography was present across all target rankings but with different rules breached.

This analysis helps prioritize organizational efforts in securing systems under development.

Rules categories not present in the top 20 reinforce the need for comprehensive testing rather than using a single technique. Furthermore, there are Rules categories present across all targets of evaluation, with different rules violated.

Categories that are not found in any top 20 reflect a lower prevalence of vulnerabilities, not non-existence.

Rules: is a set of security requirements that allows you to parameterize a pentest according to the risk appetite of your organization. Rules allows you to determine what is tested and what is not, and what is considered a vulnerability and what is not. It is also the basis for determining how rigorous a pentest was, based on tested and untested requirements. The security requirements are independent of the type of technology being used and are written as specific and understandable objectives. They are the security demands that you agree to follow and comply with. In our hacking services, we determine if these are met or not. Rules is based on several standards related to information security.

Top 20 Broken Rules: INF

#	Category	Rule	Total
1	Data	R177. Avoid caching and temporary files	1424
2	Data	R176. Restrict system objects	302
3	Authentication	R264. Request authentication	182
4	Data	R300. Mask sensitive data	156
5	Social	R261. Avoid exposing sensitive information	156
6	Logs	R077. Avoid disclosing technical information	143
7	Architecture	R266. Disable insecure functionalities	100
8	Networks	R255. Allow access only to the necessary ports	83
9	Authentication	R237. Ascertain human interaction	69
10	Data	R181. Transmit data using secure protocols	56

#	Category	Rule	Total
11	Cryptography	R150. Set minimum size for hash functions	22
12	Credentials	R134. Store passwords with salt	15
13	Credentials	R135. Passwords with random salt	15
14	Services	R262. Verify third-party components	13
15	Services	R265. Restrict access to critical processes	11
16	Data	R185. Encrypt sensitive information	10
17	Cryptography	R148. Set minimum size of asymmetric encryption	9
18	Cryptography	R149. Set minimum size of symmetric encryption	9
19	Certificates	R088. Request client certificates	8
20	Certificates	R089. Limit validity of certificates	8

Top 20 Broken Rules: APP

#	Category	Rule	Total
1	Source	R173. Discard unsafe inputs	2031
2	Architecture	R320. Avoid client-side control enforcement	1968
3	Data	R176. Restrict system objects	1355
4	Architecture	R062. Define standard configurations	1060
5	Data	R181. Transmit data using secure protocols	1059
6	Logs	R075. Record exceptional events in logs	1033
7	Cryptography	R150. Set minimum size for hash functions	979
8	Cryptography	R148. Set minimum size of asymmetric encryption	948
9	Cryptography	R149. Set minimum size of symmetric encryption	948
10	Architecture	R266. Disable insecure functionalities	913

#	Category	Rule	Total
11	Logs	R077. Avoid disclosing technical information	752
12	Authentication	R237. Ascertain human interaction	582
13	Source	R159. Obfuscate code	487
14	Data	R177. Avoid caching and temporary files	473
15	Services	R265. Restrict access to critical processes	419
16	Services	R262. Verify third-party components	410
17	Session	R029. Cookies with security attributes	317
18	Authorization	R096. Set users' required privileges	279
19	Data	R329. Keep client-side storage without sensitive data	259
20	System	R186. Use the principle of least privilege	191

Top 20 Broken Rules: CODE

#	Category	Rule	Total
1	Source	R161. Define secure default options	39710
2	Source	R359. Avoid using generic exceptions	38415
3	Source	R171. Remove commented-out code	18376
4	Source	R323. Exclude unverifiable files	14523
5	Services	R262. Verify third-party components	11086
6	Logs	R075. Record exceptional events in logs	8973
7	Data	R176. Restrict system objects	8294
8	Logs	R077. Avoid disclosing technical information	7909
9	Credentials	R134. Store passwords with salt	5203
10	Credentials	R135. Passwords with random salt	5203

#	Category	Rule	Total
11	Data	R185. Encrypt sensitive information	5203
12	Source	R156. Source code without sensitive information	3630
13	Cryptography	R145. Protect system cryptographic keys	3120
14	Data	R181. Transmit data using secure protocols	2844
15	Source	R302. Declare dependencies explicitly	1576
16	Data	R177. Avoid caching and temporary files	945
17	Architecture	R266. Disable insecure functionalities	909
18	Logs	R083. Avoid logging sensitive data	695
19	Source	R173. Discard unsafe inputs	678
20	Cryptography	R223. Uniform distribution in random numbers	636

Non-Compliance With Rules: Categories by Targets

As with the most frequent findings, we checked the most frequently breached Rules categories across targets. Six categories made it to the top in all of them:

Architecture | Credentials | Cryptography | Data | Logs | Services

Data and logs were the most prevalent.

INF

Category	Total
Data	1948
Authentication	257
Social	156
Logs	145
Architecture	110
Networks	83
Credentials	58
Certificates	48
Cryptography	41
Services	24

APP

Category	Total
Architecture	4169
Data	3447
Cryptography	2970
Source	2622
Logs	1831
Authentication	1378
Services	901
Session	783
Credentials	480
Authorization	390

CODE

Category	Total
Source	117993
Data	18345
Logs	17895
Services	11311
Credentials	11061
Cryptography	6163
Architecture	2005
Authorization	684
System	477
Social	381

Non-Compliance With Rules: Categories by Targets

It is worth noting that authentication non-compliance was found frequently in the three targets: infrastructure, applications and source code.

In turn, **authorization** was found more repeatedly in both application and source code. These violated rules are not always suitable to be tested in deployed infrastructure.

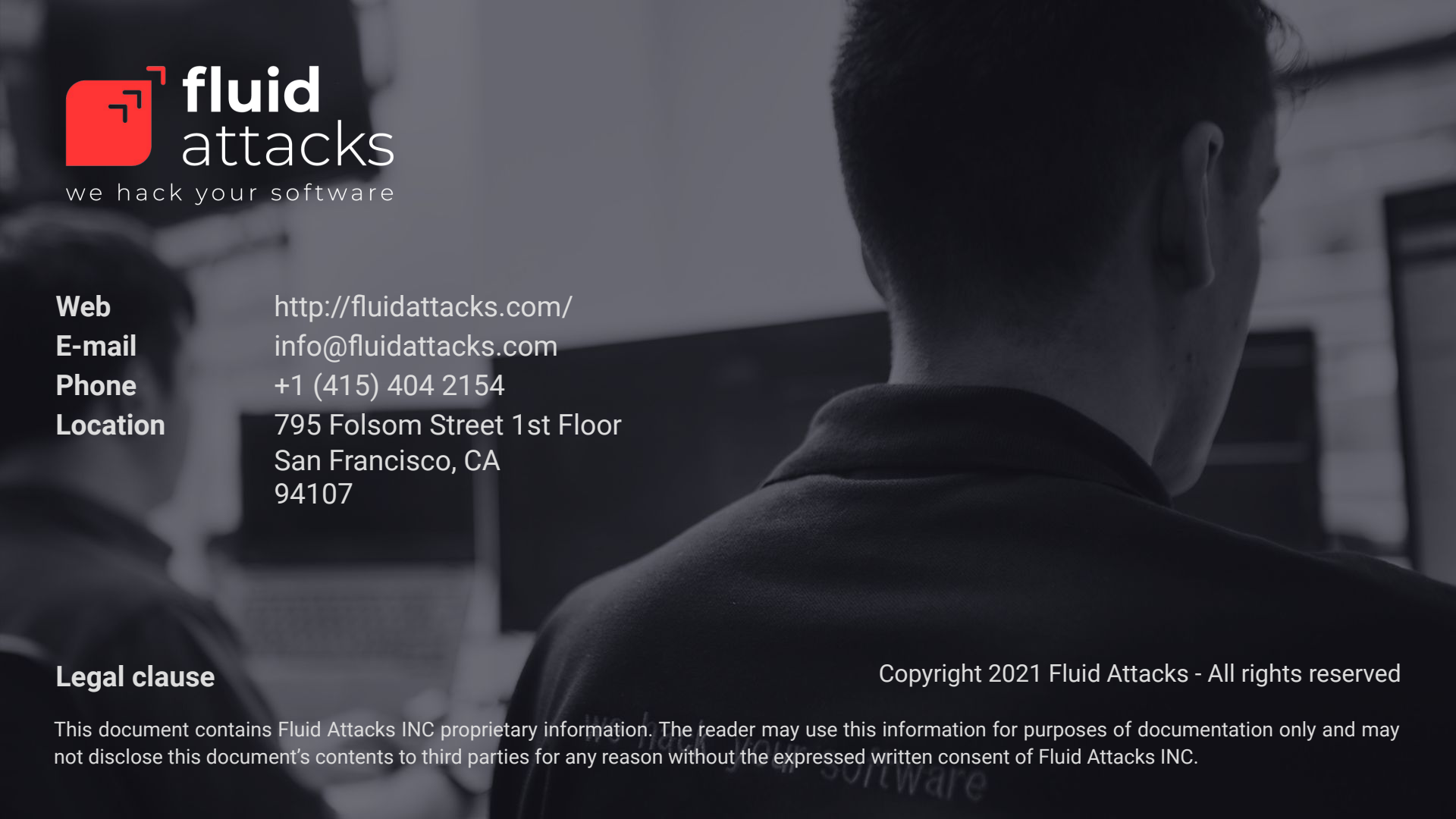
Both cases show that comprehensive testing is key to covering all potential vulnerabilities in a system. Single-technique testing is a step, but clearly not enough to assess and fix vulnerabilities in systems supporting business operations.

About Fluid Attacks

Since 2001, we have helped our clients manage their cybersecurity risks, ensuring that systems which are critical to their operations are safe for their users. Our security experts specialize in continuous hacking on apps, infrastructure and source code, covering all software development stages.

We go beyond automation. Technology is not enough to make your application secure. Fluid Attacks performs comprehensive hacking at the speed of your business, offering the combined advantages of human knowledge, automation and AI.

We provide speed, precision and scalability to hacking operations, helping companies comply with industry standards like PCI DSS, OWASP, GDPR, HIPAA, NIST, among others.



fluid attacks

we hack your software

Web	http://fluidattacks.com/
E-mail	info@fluidattacks.com
Phone	+1 (415) 404 2154
Location	795 Folsom Street 1st Floor San Francisco, CA 94107

Legal clause