State of API Security Report Q3 2022 Learn more



# Recap: The 7 Biggest API Security Incidents in 2021

Michael Isbitski

Dec 21, 2021

The world witnessed no shortage of API-related security incidents in 2021. Gartner has made a prediction on this front for a number of years, which we've seen play out precisely. Much focus has been on the Gartner strategic planning assumption (SPA) that by 2022, API abuses will become the most-frequent attack vector, leading to data breaches for enterprise web applications. If Gartner got anything wrong with its crystal ball, it's that the analysts were too conservative with the year

State of API Security Report Q3 2022 Learn more



applications will exhibit increased attack surface as a byproduct of exposed APIs rather than the user interface (or application) itself. We've seen this prediction play out as well. Web applications and mobile applications are useful as front doors to an API and reverse engineering. But the application UI is ultimately a throw-away for an attacker and exposed, back-end APIs prove to be the most valuable targets.

We've spotlighted the seven biggest API security incidents in 2021 that plagued many different companies of different sizes and across verticals. If you take away nothing else from this summary, at least realize that API security should be one of your chief concerns heading into 2022. APIs are the building blocks for systems architecture, partner integrations, and business logic. If API security strategy and supporting tooling aren't landing on your cybersecurity initiatives, you should take another look. API incidents can create significant, negative business impacts including data loss, regulatory penalties, brand damage, and privacy erosion.

## January 2021: Parler

The storming of Capitol Hill on January 6, 2021 was an unprecedented incident for American citizens and also the free world. A number of individuals used Parler, a social media platform, to socialize and coordinate their actions on that date. As a response to the event, digital activists siphoned (or scraped) as much data as possible Parler platform using the platform's APIs and prior to it being shut down. Losers are present on all sides of this event. Parler users (regardless of any proven criminal activity) experienced the worst case of privacy erosion, Parler was effectively put out of business, and the world is left with disturbing visuals of the messages and actions. Media coverage from other investigative journalists as well as reviews of the code used

State of API Security Report Q3 2022 Learn more



The shutdown of Parler was politically charged, and much of it is still playing out in the justice system, particularly with respect to the rioters. The event offers valuable learning lessons for social media services but also any organization offering a public web service that stores and presents PII. The digital activists took advantage of a number of inherent API flaws in the Parler platform to scrape data en masse. The Parler platform exhibited a number of significant flaws which included:

- Broken user authentication and/or broken object level authorization on Parler's public APIs
- Parler failed to purge user data as expected and relied on a user-controllable parameter to control visibility of deleted data
- Parler used predictable, sequential identifiers for content, which made it easier for activists to enumerate and scrape all messages, pictures, and videos in the platform
- Parler failed to scrub content metadata, which included geo-tagging in pictures and videos that gives away an individual's precise physical location at the time of posting
- Multi-factor authentication was misconfigured which potentially resulted in an authentication bypass

## What can we take away from the event?

Many media outlets dismissed the event as the result of poor coding practices, but the technical details of the flaws provide valuable API security learning lessons. The incident involves multiple parties that are both threat actors and victims. We have the Capitol Hill rioters, committing crimes against the US, who are also victims as a result of the privacy impacts from the data disclosure. We also have the activists that banded together to collect all the data, which could be viewed as malicious by Parler and the Capitol Hill rioters. Undesirable, high volume

State of API Security Report Q3 2022 Learn more

Why Salt Platform Use Cases Customers Resources Company Blo

reasonably secure, but they also must recognize that applications are just gateways to much more valuable assets: the back-end APIs that power business logic and serve data. Authentication and authorization are critically important for all system designs including APIs. However, complexity of a distributed architecture makes strong access control difficult to accomplish in practice. Organizations must still analyze behaviors of users within authenticated and authorized sessions to ensure they are not malicious API callers, such as with intent to enumerate and scrape data sets. Continuous visibility and monitoring over API consumption is critical to API security and data security strategy.

For further details: https://salt.security/blog/unpacking-the-parler-data-breach

## March 2021: Microsoft Exchange Server

Microsoft discovered and disclosed an attack campaign coordinated by Hafnium, a Chinese state-sponsored hacking group that targeted Microsoft Exchange Server 2013, 2016, and 2019 deployments in March 2021. The initial attack vector was a server-side request forgery (SSRF) vulnerability within the proxying front-end component of Exchange. The weakness allowed attackers to send arbitrary HTTP requests and authenticate as the Exchange server. Once attackers were in, they could exercise a more complex attack chain to gain access to the operating system itself, execute code remotely, and pivot in their attack to other systems within an organization's network.

### What happened and why is it noteworthy?

The initial entry point for the attack was through the Client Access Service (CAS) component of Exchange Server 2013, 2016, and 2019. CAS functions as a proxy for front-end clients to interact with Exchange

State of API Security Report Q3 2022 Learn more

Why Salt Platform Use Cases Customers Resources Company Blo

attack chain. An attacker could submit an unauthenticated HTTP request to someone's Exchange server infrastructure by overloading cookies that controlled authentication within Exchange itself. The closest OWASP API Security Top 10 mappings for this specific issue would be broken user authentication and security misconfiguration. The Microsoft Exchange instance in turn processes any submitted commands embedded in the web request within the back-end APIs of Exchange. Attackers could exploit a set of four vulnerabilities to inject commands into a web request, escalate privileges, maintain persistence by planting a web shell, and attack other systems within an organization's network. This other aspect of the attack chain would map closest to OWASP's definition of injection. The attack chain traverses the web channel down to application binary and operating system level.

Microsoft reacted quickly to try and contain the damage to customers who maintain their own Microsoft Exchange instances on-premises or in private cloud. Microsoft provided a number of tools and scripts to aid Exchange admins in determining if their infrastructure was vulnerable at all, mitigating the initial attack vector as appropriate, and identifying if attackers had already compromised servers with planted web shells. The remediation scripts look for those crafted HTTP requests in server logs and the subsequent binary and operating system commands attackers would have to execute to maintain persistence.

### What can we take away from the event?

The impact of the incident was substantial resulting in a number of government notifications and advisories. Quite a few enterprises operate their own mail infrastructure rather than consuming mail services as SaaS with Microsoft Office 365. Microsoft Exchange is ultimately a collection of many moving parts including front-end

State of API Security Report Q3 2022 Learn more

Why Salt Platform Use Cases Customers Resources Company Blo

The age-old security principle of making sure your systems and applications are patches applied here. However, exposed and vulnerable Exchange instances continue to persist on the Internet. Vulnerable Exchange instances can be compromised and used maliciously in a number of ways, some of which Microsoft has observed and documented as part of its postmortem analysis. The possibilities include ransomware attacks, credential theft, cryptocurrency mining, and repurposing for use as botnets for perpetuating attacks on other systems or networks.

The initial attack vector requires the ability to make an untrusted connection to the Exchange server, but this is unavoidable with Exchange instances and front-end components so that users can authenticate into their email services. Those front-end elements must be partially exposed to enable remote work and mobility with distributed workforces. Most Exchange instances would not restrict untrusted connections or mandate use of a controlled network connection such as with VPN since it can conversely inhibit productivity. Such network security measures also only protect against the initial attack vector. API communications within Exchange infrastructure and to other systems are often unmonitored and unprotected.

For further details:

https://www.microsoft.com/security/blog/2021/03/25/analyzing-attacks-taking-advantage-of-the-exchange-server-vulnerabilities/

https://devco.re/blog/2021/08/06/a-new-attack-surface-on-MS-Exchange-part-1-ProxyLogon/

State of API Security Report Q3 2022 Learn more

Why Salt Platform Use Cases Customers Resources Company Blo

called an API that was hosted by Experian which was designed to assess an individual's credit worthiness as part of promotional inquiries. Third-party lending sites, such as what Bill was using, would call the Experian API to validate user-provided PII including first name, last name, address, zip code and birthdate. A classic definition of leaky API, the Experian API used minimal information to identify and authenticate an API caller, and in return the API served personal data in the response.

### What happened and why is it noteworthy?

The credit information returned by the Experian API included FICO scores and risk factors that impact the given individual's credit history including number of credit accounts, proportion of balance to available credit, and age of credit accounts. The API was designed to use minimal identifying and authentication material, most of which is essentially public. The required authentication material can be gathered through Internet searches of public repositories or by repurposing data from other breaches.

The Experian API exhibited the following weaknesses:

- Weak or broken authentication since the API required minimal authentication material to access and retrieve data
- Security misconfiguration because the API did not properly validate one of the key identifiers, birth date, and could simply be fed zeros
- Excessive data exposure since the API provided sensitive data in the form of FICO scores and credit risk factors with minimal, easily guessable, or brute forceable authentication material
- A likely lack of resources and rate limiting since the researcher also

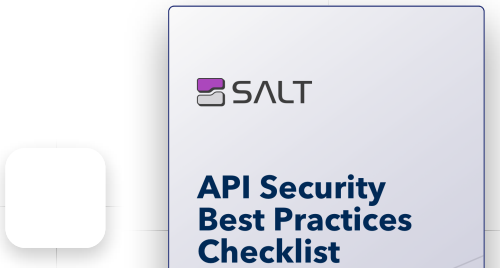
State of API Security Report Q3 2022 Learn more



The Experian API likely needed to be architected in such a way that data would be easy to obtain by third-party lenders to facilitate lending applications. This design challenge is a common dilemma with many public or open APIs that must use minimal or no authentication. Traditional security approaches that promote network access control, authentication, and authorization quickly fall over.

If Experian had implemented strong access controls, they would've complicated integration with partners and likely cut into potential business. It's also likely that the number of financial institutions that Experian operated this API for is more than they could restrict using IP address allow or deny lists. IP address allow/deny lists as well as rate limits are often promoted as security best practice. However, such controls don't scale for mass API consumption, unpredictable number of callers, or distributed API callers.

Ultimately, organizations need to continuously monitor API consumption from customers and also through partner and supplier integrations. This feat can't be accomplished manually given the amount of API telemetry that must be collected to establish baselines and identify anomalies like an abusive API caller. The nightmare scenario here is that an attacker could enumerate and scrape the data held by Experian. Leaking credit worthiness data for one individual may be dismissed as a non-event, but if a malicious party were to grab this information for a large chunk of the population it creates massive




Get the free customizable API Security Best Practices Checklist

State of API Security Report Q3 2022 Learn more

Why Salt Platform Use Cases Customers Resources Company Blo

## April 2021: John Deere

The security research firm, Sick Codes, discovered and disclosed vulnerabilities about leaky APIs operated by the agricultural equipment manufacturer, John Deere. The company is well known for producing tractors, harvesters, and other farm equipment. The push to connect and computerize physical product offerings as part of Internet of Things (IoT) efforts is very prevalent. In IoT implementations, any number of devices frequently relay data via web APIs to some central service or back end. Some functionality of IoT devices, such as firmware updates or adding new functionality, may also be powered by similar web APIs. In the case of John Deere, some use cases include device updates pushed from a central website, farming automation, and GPS auto-steer.

### What happened and why is it noteworthy?

The security researchers discovered that it was possible to call John Deere APIs and determine if a particular username was taken. They automated calls against this API using a pre-existing list of Fortune 1000 company names and were able to identify which companies were John Deere customers and their associated account names. The API could be called without any authentication material, which the researchers discovered by stripping out all cookies on requests and still succeeded at retrieving data. The API endpoint also had no rate limits in place, so the researchers were able to enumerate the entire data store in less than two minutes and determine that roughly 20% of the Fortune 1000 companies had John Deere accounts.

The researchers also discovered another API endpoint that was part of

State of API Security Report Q3 2022 Learn more

Why Salt Platform Use Cases Customers Resources Company Blo

owner, physical address registered with John Deere, other tractor equipment identifiers, remote accessibility, and subscription start and stop dates. The John Deere Operations Center API also returned data about third-party equipment and was not limited to just John Deere products. VINs can be readily obtained from general auction sites like eBay or tractor-specific auctions sites. This particular API endpoint did require authentication, but it didn't properly authorize API callers. It was possible for the researchers to authenticate using a developer account created within the John Deere portal and query other VINs for which they were not the owners.

### What can we take away from the event?

Leaky APIs and the ability to scrape them is at best limited to brand damage, which is never a simple matter to assess. Worst case, leaky APIs give attackers valuable information they can use to perpetuate other attacks and fraud. Organizations may also be subject to regulatory penalties depending on the vertical or geographical region they operate in. Organizations must also consider the citizenship of the individual that transacts with their services as seen with privacy regulation like GDPR.

Organizations that build or integrate APIs must also have consistent, documented vulnerability disclosure processes and/or supporting programs. The Sick Codes security researchers ran into a number of hiccups and lost time trying to report the issues to John Deere. Even something as simple as identifying where to send technical details wasn't obvious. It's not uncommon for organizations to lack resources to handle intake and triage of bugs or vulnerabilities reported by external parties. This security program gap is where crowd-powered bug bounty platforms can help, but not all organizations make use of e services. In this case, John Deere stood up a VDP on HackerOne

State of API Security Report Q3 2022 Learn more



counterparts. Risks are amplified though and can also include safety concerns. 2021 brought some renewed focus on digital supply chains now as part of cybersecurity programs. The security researchers expressed concerns that the API exposure could have jeopardized the food supply chain in the United States. The APIs provide data on equipment that few would argue is critical to maintaining agricultural output and feeding a population.

For further details: https://sick.codes/leaky-john-deere-apis-serious-food-supply-chain-vulnerabilities-discovered-by-sick-codes-kevin-kenney-willie-cade/

## May 2021: Peloton

A security researcher with Pen Test Partners, Jan Masters, found that he could make unauthenticated requests to Peloton back-end APIs that were used by related exercise equipment and subscription services. One could call the Peloton API endpoints directly and obtain significant amounts of PII, resulting in privacy impacts for Peloton customers. The PII included user IDs, location, weight, gender, age, and more. Peloton web and mobile applications created as companions to Peloton exercise equipment used these back-end APIs to provide workout statistics and class scheduling. Peloton ultimately remedied the API issues, but it's unclear how many Peloton customers may have had their personal information disclosed.

### What happened and why is it noteworthy?

The incident spotlights modern application design principles, specifically mobile app front ends interacting with back-end APIs to provide data and functionality. Back-end APIs often lack appropriate authentication and authorization since application teams architect with the notion that API consumption will flow through the expected and

State of API Security Report Q3 2022 Learn more



and API call sequences with the ultimate goal of exercising and abusing the back-end APIs directly.

The leaky APIs from Peloton exhibited the following weaknesses:

- Broken user authentication since initially Peloton had no authentication in place on multiple endpoints including work out details, user details, and GraphQL data access APIs. Any person with Internet access could query Peloton APIs directly and obtain PII on Peloton users.
- Broken object level authorization which materialized when Peloton attempted a fix early in the vulnerability disclosure process. Access required authenticating with a Peloton account, even those with a basic subscription without exercise equipment. Once an attacker obtained an authenticated session, they could query for any other user's PII since no authorization mechanism was in place.
- Excessive data exposure resulting from the API returning a number of PII in responses, relying on the front-end mobile application to filter out sensitive data appropriately.

### What can we take away from the event?

Organizations should always assume their APIs are a known quantity to attackers, and those APIs will be targeted directly. Attackers find and abuse API endpoints by analyzing application traffic and reverse engineering front-end code. Mobile or IoT channels provide little to no obscurity over web channels, and attackers move between channels as appropriate to perpetuate their attacks

As we see in countless API incidents, authentication and authorization are problem spots for many organizations. Access controls are difficult to implement in distributed architectures that serve multiple types of consumers that may be users or machines. Modern designs need to

State of API Security Report Q3 2022 Learn more



Security Top 10. Organizations should always authenticate API callers. They should continuously validate authorization of authenticated users to ensure they are entitled to given data or functionality. And they should also monitor user sessions continuously for potentially malicious behaviors in cases where an API caller is abusive.

For further details: https://salt.security/blog/the-peloton-api-security-incident-what-happened-and-how-you-can-protect-yourself

## June 2021: LinkedIn Scraping

Two cases of this incident popped up in 2021, first in April 2021 and then again in June 2021. A large repository of private data had been posted for sale on an underground marketplace, RaidForums. The data set contained information on roughly 500 million individuals at first but eventually ballooned to 700 million. It was constructed by scraping and correlating data from a number of websites, with a substantial chunk being LinkedIn profile data.

### What happened and why is it noteworthy?

Microsoft maintained that the data was publicly viewable profile information that may have been scraped from the social media service, but it did not constitute a data breach. The leaked data set included a host of personal data including full names, LinkedIn profile names, LinkedIn IDs, LinkedIn profile URLs, gender, email addresses, phone numbers, physical addresses, geolocation records, industry information, personal experience, and professional experience.

Microsoft went further to state that any misuse of member data, including scraping, is a violation of the platform's terms of service. They iterated that no private LinkedIn profile was part of the leaked data et they were able to review. Microsoft also stated that they "work to

State of API Security Report Q3 2022 Learn more



The complete data set totaling over 100GB for 700 million users was eventually leaked publicly in September 2021. Most of the data was indeed public, with the exception of email addresses. The correlated data set, specifically full name matched to email address, can be valuable to attackers who might use the information to enrich other data sets, perpetuate other fraud, or target users directly with phishing, ransomware, and social engineering attacks. Large, compiled data sets like this one and others like COMB can be useful to attackers in credential stuffing and brute forcing attack campaigns with the goal of achieving account takeover (ATO).

### What can we take away from the event?

While the incident was quickly dismissed as a non-event or "not a data breach," it's an interesting examination of the state of privacy. Clearly, if the data set consists largely of LinkedIn data, the attacker was able to succeed in their scraping efforts and build the repository. We've seen many cases of scraping via APIs, which provide a mixture of both public and private information. The more data you can aggregate and correlate on individuals, the quicker privacy starts to fall over. Data collected and analyzed at scale allows for inference, which can present problems when organizations are trying to adhere to privacy regulation like GDPR.

Microsoft is not alone here. We've seen similar incidents of data exposure or leaked personal data with other social media services like Facebook and Instagram. Aggregated and correlated data is a valuable resource to attackers. The way they assemble such data is typically by automating requests against APIs and enumerating through records to piece together the larger data set. If an API uses weaker authentication or authorization, or it serves excessive amounts of data in responses, it's an even bigger target for attackers. Strictly speaking and by most regulatory definitions the event doesn't constitute a data breach. But

State of API Security Report Q3 2022 Learn more

Why Salt Platform Use Cases Customers Resources Company Blo

https://fortune.com/2021/04/08/linkedin-user-data-breach-leak-hackers/ and

https://fortune.com/2021/06/30/linkedin-data-theft-700-million-users-personal-information-cybersecurity/

## December 2021: Log4j

News of a vulnerability called “Log4Shell” or “LogJam” broke in December 2021. It’s formal designation is “CVE-2021-44228,” and it quickly gained worldwide attention because of its prevalence and the severity of the security risk. The vulnerability was first reported by Chen Zhaojun of the Alibaba Cloud Security Team, and it quickly impacted a number of large services including Amazon.com, Apple iCloud, Cloudflare, Twitter, and many other victim organizations.

### What happened and why is it noteworthy?

The Log4Shell vulnerability is one of the most dangerous and most widespread vulnerabilities ever discovered. The vulnerability exists within the popular Java logging library from Apache, log4j, which many Java-based applications rely on for logging capabilities. Java powers many critical devices, systems, and services, as can be seen by the number and variety of impacted organizations.

In the case of Log4j, user supplied input may not just be logged to disk as most would expect in a logging mechanism. Input data is fed to and processed by lookup plugins. With Log4Shell specifically, the Java Naming and Directory Interface (JNDI) lookup plugin is where problems arose. JNDI queries should normally only include object names, and lookups are intended to occur on predefined servers without user-controlled input. However, the Log4j context lookup function allows for

State of API Security Report Q3 2022 Learn more



Log4j, and in turn the system would make a request to that resource. This resource could be an attacker-controlled server where arbitrary, malicious code lives that the system would in turn execute. The vulnerability could also be leveraged by attackers to access other internal servers and resources. Salt observed and continues to see many Log4j exploitation attempts against customers within our platform.

### What can we take away from the event?

The event shares some similarity with other security incidents like the Heartbleed issue that plagued OpenSSL in 2012 or Apache Struts issues that surfaced in 2017 and led to the Equifax breach. These high-profile security events have a consistent theme of component or dependency re-use. Organizations regularly build infrastructure, applications, and APIs by repurposing open-source componentry. Open-source component use speeds up development and delivery time, and it can help with introducing security frameworks or secure defaults. Conversely, if a dependency is found to contain a latent vulnerability, it can create significant upstream and downstream impacts. The realities of nested or transitive dependencies also quickly create a complex spider-web of inherited bugs and vulnerabilities.

Dependency analysis and software composition analysis (SCA) tools can help with finding and fixing some dependency problems. Still, problems arise such as who owns the code to create a fix, or what is a “good enough” measure for quality of a component. Many dependencies have latent vulnerabilities of lower severity that may still be usable in production. The commonly available dependency analyzers and SCA scanners also only address the problem in the context of binaries and componentry. Web APIs are another form of application dependency which these tools just don’t cover. Most organizations

State of API Security Report Q3 2022 Learn more



Security teams operating WAFs and API gateways are likely unable to create rules or message filters quickly enough to cover all the possible mutations of the Log4j JNDI injection patterns that keep arising. Organizations may also be relying on their WAF vendor to handle rule creation automagically, which infers they'll be unprotected until the vendor can create and deploy an effective rule to all their customers. Worsening matters, organizations commonly deploy these controls to protect only Internet-facing assets and not to mediate internal API calls where Log4j issues can present themselves. Limited deployment of production security controls is a side effect of risk-based prioritization, constrained resources, and restricted budgets that all security programs must contend with. Organizations need a holistic view of API calls at multiple points of their enterprise architectures that is aided by machines to analyze all those signals and help discern malicious API callers from benign ones.

For further details: https://salt.security/blog/the-log4shell-cve-2021-44228-vulnerability-what-it-is-how-it-works-and-how-to-protect-yourself

Tags

API Attack Analysis API Attack Prevention API Security 101

API Vulnerability Analysis

Subscribe to Our Blog

Go back to blog

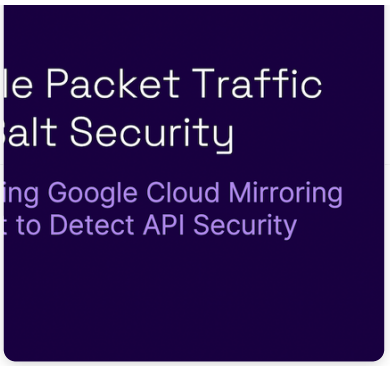
State of API Security Report Q3 2022 Learn more




## Salt Joins the Azure Marketplace and is Co-sell Ready!

Our award-winning Salt Security API Protection Platform has been added to the Azure Marketplace and we have achieved Azure IP Co-sell Ready status.

Gilad Barzilay
November 9, 2022




## Leveraging Google Cloud Packet Traffic Mirroring with Salt to Detect API Security Threats

It's not enough to find and block attackers exploiting a vulnerability in your API. You will also want to remediate the security gap in your APIs.

Stephanie Best
November 4, 2022


## OpenSSL Vulnerability Analysis – Denial of Service and Remote Code Execution

Insights regarding two new vulns that have been uncovered in the OpenSSL library – CVE-2022-3602 and CVE-2022-3786. These vulnerabilities affect OpenSSL.

SALT LABS

Salt Labs
November 3, 2022

**State of API Security Report Q3 2022** Learn more

Why Salt | Platform | Use Cases | Customers | Resources | Company | Blo

3921 Fabian Way
Palo Alto, CA
94303

+1 (650)
254-6580

**Contact us**

Platform

Customers

Resources

Blog

Discover all your APIs

Prevent sensitive data exposure

Stop API attacks

Prevent account takeover, data exfiltration

Accelerate incident response

Provide remediation insights

Simplify compliance

Salt

Salt Labs

Newsroom

Events

Careers

Terms of Use Privacy Policy Copyright © 2022 Salt Security