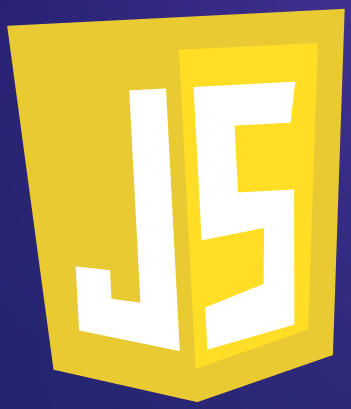
feroot

FEROOT.COM



The Ultimate Guide to

JavaScript Security

Learn everything you need to know about JavaScript security to protect your web applications and websites from cyberthreats.

Contents

01

Introduction

The complexities at the heart of modern websites and web applications often begin and end with a programming language called JavaScript (JS). JavaScript enables common web programming features, such as storing variables, creating operations on texts, and running code based on certain web events. These features are key drivers in e-commerce, financial transactions, and data input. JavaScript impacts many things in our daily lives, from transferring money from one account to another, to ordering products online, and to watching how-to videos.

JS is the predominant and essential code that drives web applications. It is also fairly easy to learn. Unfortunately, the ease of coding in JavaScript also gives it a low barrier to entry, enabling beginners to build fractured and vulnerable code, and developers with a more malicious intent to leverage flawed code for their own evil purposes.

The important thing to understand about JavaScript code is that threat actors and attackers actively search for vulnerabilities to exploit, in even the most well-written code. And increasingly, they're looking specifically for vulnerable JS code on the 'client side.'

Client-Side JavaScript Web Application Security

Modern websites, the majority of which are written in JS, and the software that drives them, carry risk. Any customer—be it a consumer or another business—that wants to conduct a transaction via a website is going to expect a seamless and safe user experience with minimal or no risk. The first step in protecting customers is making sure the entry point into your business—your JS web applications and webpages—is as secure as it possibly can be.

In this e-book, we offer businesses a comprehensive look at how to secure client-side JS applications and the type of attacks that are increasingly targeting businesses that deliver products and services through the client side utilizing JS applications. The security gap around the client side is growing and organizations need to be prepared to secure their front-end operations if they want to ensure business growth and customer safety. JavaScript-based websites and applications are unique and require a specific and dedicated security approach. Protecting users that operate in a JavaScript environment means understanding and acknowledging the risks and taking appropriate action to protect users and businesses.

Know Your Terminology: Client Side vs. Server Side

Today, **98%** of webpages and web applications are written in JS. These applications are found on the front end or the client side. Before diving into JS security, let's level set to make sure we understand the core JS security terminology.

What Is the Client Side?

Within the context of IT, the words "client" and "server" relate to the basic structure of connected devices. So, end-user devices, like desktops, laptops, mobile phones, and tablets are considered 'clients,' while the systems that the devices are connected to are referred to as 'servers.' For web developers, the term 'client side' means any activity that is taking place on the client device. This could be a webpage as it is being displayed, including text, videos, and images, or any operation or calculation underway in the background.

vs

What Is the Server Side?

Server side refers to anything happening on the server instead of the end user's device. In the short span of internet history, there was a time when everything operated on the server side, even things like dynamic webpages. However, the process of transmitting data from the end user's device (client side) to the server side took far too long—something referred to as latency. To combat this and other problems related to monolithic architecture, web developers started building code (sometimes called scripts) that operated more on the client side, making webpage operations much faster.

What Is JavaScript Security and Why Is It Important?

Client-side security or JS security refers to the technologies and policies used to protect an end user from malicious activity that is occurring on dynamic webpages accessed from the end user's own device. JS security is a practice by which businesses protect their end users from incidents, vulnerabilities, and attacks that occur within an end user's browser. The browser is also sometimes referred to as the "front end" in the context of code development for web applications. JS attacks have been increasing in both scale and cost since the beginning of 2020 as companies expand their investment in the end-user digital experience. This has created an unprecedented opportunity for threat actors to exploit end-user activities.

To fully manage the risk against breaches and attacks, companies must simultaneously protect both the client side and back end of their application portfolios. This includes everything that the customer sees, such as text, images, stylesheets, and the rest of the user interface, along with anything the customer interacts with, such as what the website or web application does within the user's browser.

One of the most important actions any business can take is protecting their customers from JS threats. Unfortunately, because of the sophisticated and subtle nature of these attacks, they can be hard to detect until it's too late. To ensure that businesses are offering a safe and secure digital experience, they must be diligent about securing their website and web applications from dangerous client-side JS attacks.

02

Client-Side Attacks and JavaScript Code

What is JavaScript?

JS is a text-based programming language used on both the client side and server side. JS allows developers to build websites and web applications with interactive elements that engage the user and enhance their experience. JS is relatively easy to learn and easy to share.

More than

98%

of websites use JavaScript for client-side webpage behavioral elements.

JS serves as one of the core technologies used to build web applications and websites accessed by consumers. Over 98% of websites use it for client-side webpage behavioral elements. Eighty percent of websites use a third-party JS library or web framework for their client-side scripting. What this means is that websites are assembled with various pieces of third- or fourth-party JS code, which do not have any built-in security permissions. JS is inherently vulnerable to cyberattacks. JS allows threat actors to deliver malicious scripts to run on a client computer via the web.

Developers and marketers love JS because of its flexibility and ease of use. Security teams are nervous about JS, because it does not have "security permissions" built into it and is difficult to keep safe from client-side focused threat actors.

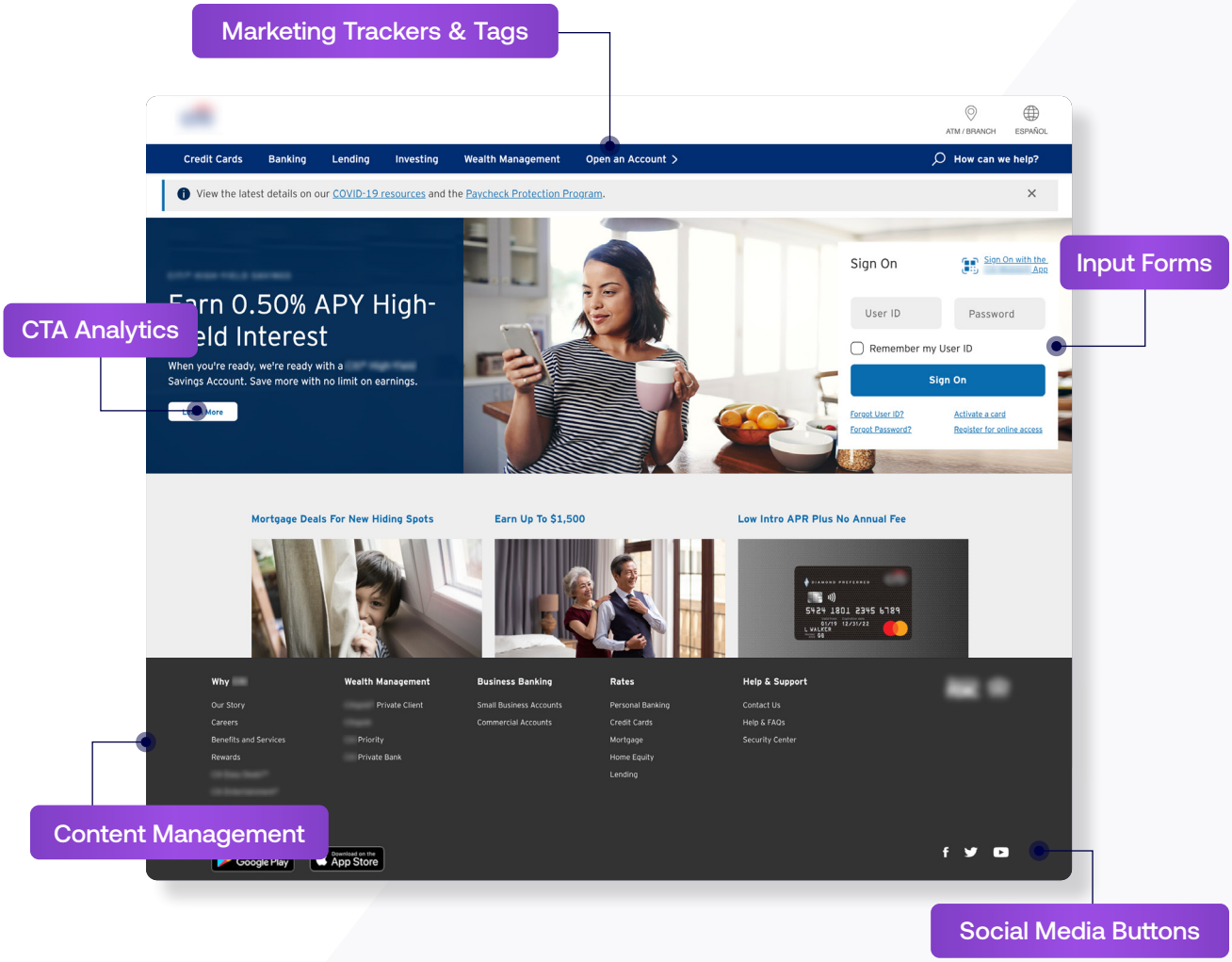
What Is the Relationship Between the Client Side and JavaScript?

JavaScript is the most commonly used client-side process

Many webpage operations that an end user sees are actually happening only on the end user's device and not on the server. Web developers achieve this by creating code within the web application to annotate or format text on a webpage. This code is called "markup language" (e.g., HTML).

Web developers also use something called 'client-side processes' in their application architecture. Client-side processes enable dynamic webpages to operate on the client device, instead of the server. (Dynamic webpages display different content depending on user input.) JS is the most commonly used client-side process. Markup languages and client-side processes enable the web developer to build an interactive web application architecture that responds to something that the end user is doing on the client device. For example, if a person is shopping on a website, a client-side process would recognize the end user's mouse moving over to a form and clicking on a blank space to fill in a credit card number.

Businesses are putting too much faith in the security of the dynamic supply chain

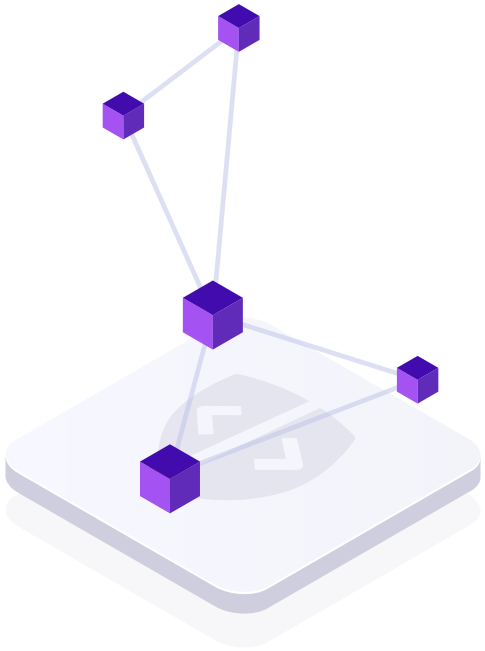


Client Side

- CDNs, APIs
- Websockets
- XHR/Fetch
- Open Source
- Documents
- Stylesheets
- Scripts
- Images

Server Side

- Database
- Content
- APIs
- Static Assets
- Storage
- Libraries
- Open Source



What Does JavaScript Have to Do With Security?

Because client-side activity happens outside of a business's security perimeter, typical security technologies will not protect the end user from malicious activity that is occurring on dynamic webpages accessed from the end user's own device. So, let's say you operate a small online shop that uses JS. A threat actor has injected some malicious code into your JS that enables them to capture a customer's credit card number when the customer types it into the payment page. Your standard security will not see this happening because it is not happening on your server. Instead, it is happening on the dynamic webpage that your customer sees and interacts with. Essentially, your customer has downloaded malicious code from your server, which is then interpreted and rendered by the user's browser on the user's device.

Recent research suggests that web application attacks account for more than a quarter of all data breaches. Since many web applications are written in JS and operate on the client side, your customers become vulnerable to attacks like Magecart, e-skimming, side loading, cross-site scripting (XSS), and formjacking. But vulnerable web applications are not the only way that threat actors access client-side assets. Server-side website security misconfigurations and the addition of third- and fourth-party code or supplemental applications can also cause security problems.

Therefore, implementing effective client-side security is critical if you want to protect your customers.

Why Is JavaScript Vulnerable?

JS is vulnerable because it is easy for hackers and other threat actors to input query strings into forms to access, steal, or contaminate protected data. JS is the standard for the processing of personal information in client-side websites and applications. There are numerous open-source and third-party libraries available today that include solutions written in JavaScript. These libraries offer easy access to threat actors and often contain code with known vulnerabilities. Malicious or vulnerable third-party code can provide a threat actor with unrestricted access to sensitive data at the browser level, so the attack surface is broad and wide open.

By default, JS environments do not have a security permissions model built in. The World Wide Web Consortium standard is that security permissions (what code is able to execute and what script activity is allowed) are housed in browsers, and the responsibility to manage them lies with the site or application owner. **Therefore, the responsibility is on site owners (and not the end user) to implement policies and procedures to detect and mitigate JS-borne cyberattacks.**

Why Is Front-End Business Logic Becoming More Prevalent?

The front end, also known as the client side, is what facilitates the digital user journey. It is the common ground for modern-day business logic. Why? Performance. The fewer requests made to the server, the more performant a web application can become. By utilizing JS libraries and frameworks, developers can build modern digital experiences directly in the browser.

Unfortunately, developers are not the only team delving into the client side. Threat actors are taking a deep look at the front-end business logic to see how they might be able to deface JS applications, deploy malware, or weasel their way into the back end to cause damage.

Client-Side Data Storage

One area where web applications have not changed much is with the utilization of databases. Using a back-end data store allows web applications to make queries and render the selected data into the user's view (e.g., browser).

Client-side data storage works in a similar way, but instead of being stored in the back end, the data is stored within the browser. JavaScript can then be used to access the browser directly (no back end required). This allows businesses to develop and maintain fantastic web applications and websites that deploy many unique features to enhance the user experience, including:

- Personalized site preferences for the user based on previous data to enhance the user's personal experience.
- Maintaining previous website activity and interactions to make it easier for the user to do business (e.g., remembering shopping cart contents or keeping the user logged in).
- Storing data and web assets locally so the site will load quicker for the user.
- Saving documents generated by the web application locally so that the user can still read the contents even when offline.

The Risk of Leveraging Third Parties for Innovation

Historically, websites were coded line by line, by one developer or one development team, usually in pure HTML/CSS/JavaScript without the use of pre-written code from third-party libraries. Now, they are assembled. Large chunks of prewritten code are appropriated from open source communities like Github. Themes, scripts, templates, plugins, and blocks of code from disparate sources are commonly pulled together to make a website. In fact, today, web applications load an average of over 20 third- and fourth-party scripts as part of the user experience. While third-party code and applications can quickly add functionality, they often contain vulnerabilities that can be easily manipulated by threat actors. WordPress is the world's largest web platform, with tens of thousands of ready-to-use plugins, available as easily as hitting the download and install buttons. These components load hundreds of JS scripts from all over the world.

Many web application development teams use third-party plugins and scripts for convenience purposes. It's easier and much more cost effective to deploy a tried, true, and tested script than write something similar from scratch. The greatest benefit of using third-party scripts is being able to take advantage of the creativity and ingenuity of other developers who are not part of your immediate development team. The application development community is filled with brilliant minds all over the globe who have great ideas, build innovative scripts, and then share them with their community on open-source platforms. There are thousands of developers out there who feel intimately connected to their work and their community, and nothing makes them happier than helping their colleagues build cooler web applications. Developers are just awesome like that. They love to build interesting stuff and make it available for others to use.

As mentioned earlier, third- and fourth-party code is often accessed from open-source repositories like GitHub. And herein lies the issue. When you bring in somebody else's JS code, you not only incorporate the code's logic, but you also incorporate everything else the developer has embedded —including vulnerabilities and coding errors. When building a website using third-party code, you are trusting the developer to have considered application security while they wrote it.

Unfortunately, for some developers, security is an afterthought, so businesses risk inadvertently introducing vulnerabilities, code weaknesses, and, worse, malware into their web app or website.

Another downside to leveraging third parties for innovation is that their scripts might be infected with spyware and keyloggers purposefully built to steal sensitive data without the company's knowledge. If installed, this malicious JS code can lie undetected and seemingly benign, while performing countless nefarious acts. For instance, It can scrape customer info from a form field, so when someone is entering login credentials, the malicious code is also mirroring and capturing their data. It may do the same thing on payment pages or chatbots—essentially anywhere the user is keying in information to interact with the website. **So leveraging third parties is a gamble. With a roll of the dice, you are taking risks on the safety of the third-party scripts.**

Client-Side JavaScript Timeline

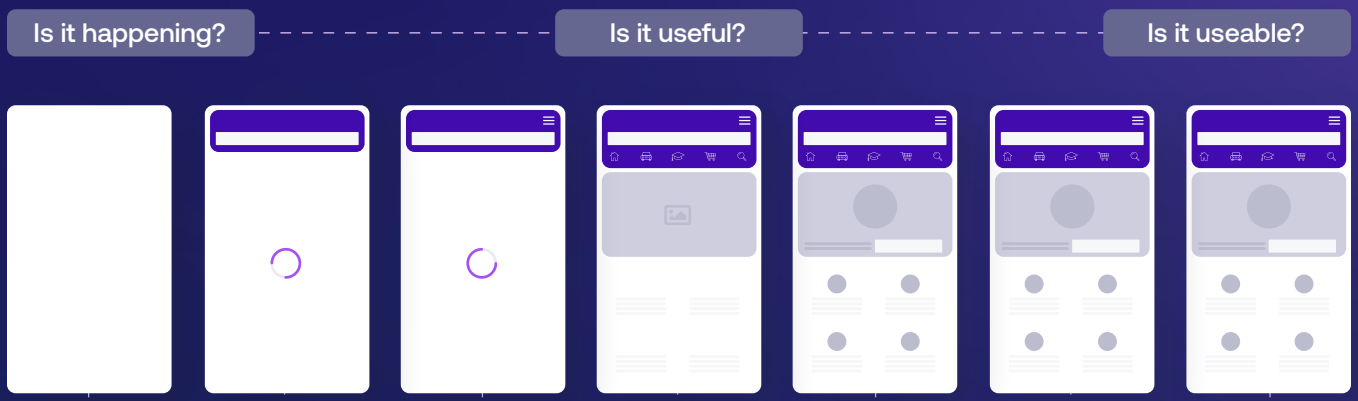
To understand how JS web applications load in a user's browser, let's take a look at the webpage assembly process and an example of how JavaScript is used to enhance functionality for specific users.

To secure JS web applications, developers need to understand the points at which threat actors can manipulate the standard JS webapp loading process.

The Webpage Assembly Process

1. A user types a URL into their browser.
2. The web browser creates a document object.
3. Request/load data from server.
4. The web browser begins rendering the webpage by adding element objects and text nodes to the document object. The *document.readyState* property is "loading" at this stage.
5. If the HTML parser encounters *<script>* elements that don't contain *async* or *defer* attributes, the parser adds those elements to the document. The inline or external script is then executed.
6. The system executes these scripts synchronously, and then the parser pauses while the script downloads and runs.
7. If the desire is to insert scripts, *document.write()* is entered into the input stream, with this text becoming a component of the document when the parser begins again. The purpose of synchronous scripts is to define functions and register event handlers for later use. When synchronous scripts are run, they can also traverse and manipulate the document tree, meaning these scripts can see their own *<script>* element and document any content that exists prior to it.
8. Should the parser encounter a *<script>* element that has the *async* attribute set, it begins downloading the script text and continues parsing the document. Once the script has downloaded, it will be executed, however, the parser continues to run. (NOTE: Asynchronous scripts do not use the *document.write()* method since the scripts cannot see their own *<script>* element and all document elements that come before it. They also may not have access to additional document content.)
9. With document parsing complete, the *document.readyState* property changes to "interactive."
10. At this point, scripts with the *defer* attribute set are executed, in the order in which they appeared in the document. Async scripts may also be executed. (NOTE: Deferred scripts have access to the complete document tree and must not use the *document.write()* method.)
11. Next, the browser fires a *DOMContentLoaded* event on the document object to mark the transition from synchronous script execution phase to the asynchronous event-driven phase. There may also still be async scripts that have not yet executed at this point.
12. While the document is entirely parsed at this point, the browser may still be waiting for additional content to load (e.g., images). When all scripts, including async scripts, have loaded and executed, the *document.readyState* property changes to "complete." The web browser now fires a load event on the Window object.
13. Moving forward, activities such as user input events, network events, and timer expirations will asynchronously invoke event handlers.

The Webpage Assembly Process



Navigation Begins
Time to first byte

First Paint
The first non-blank paint on screen

First Contentful Paint
Navigation has successfully started

First Meaningful Paint
Page's primary content is visible

Visually Ready
Page looks nearly done

Time to Interactive
Visually usable and engagable

Fully Loaded
End of load lifecycle

03

Securing JavaScript

Code Reviews

As any developer knows, code reviews are one of the most manual tasks, particularly if they're taken seriously and done the right way. In a code review, members of your web application development team will read the code that other developers wrote to evaluate its effectiveness and to look for any potential vulnerabilities. The code review process should never be skipped. Businesses also need to have strict processes and procedures in place for code reviews, to ensure vulnerabilities and code inconsistencies are noted and resolved quickly.

The effectiveness of code reviews depends on how closely developers examine each other's code. If a developer is just scrolling through code to "check the box" and complete the code review quickly, then that developer is opening up a business to high levels of risk. To optimize code reviews, organizations should use a linter tool to automatically analyze the code for common mistakes, errors, bugs, or suspicious constructs. Using linting also forces developers to write cleaner code from the onset, which in turn makes it easier for other developers to review code properly and uncover issues.

Finally, businesses need to either hire developers who have a security background, or they need to train their developers to understand the basics of security. This is even more critical if your organization uses web applications to capture and share sensitive data. Developers need to know which client-side threats and vulnerabilities they need to look for in their applications, so that code can be repaired or patched accordingly.

Threat actors love to attack vulnerable client-side shadow code for:

- Malicious code injection
- Magecart attacks
- E-skimming
- Website defacement
- Data exfiltration or compromise of sensitive organizational or customer data
- Watering hole attacks (attacking users that visit your website)
- Script attacks
- Ad injections
- Clickjacking
- Side loading
- Malicious code insertions
- Cross-site scripting (XSS)

For more information on these threats and vulnerabilities, please visit the JavaScript Risks and Threats section of this e-book.

Authentication and Authorization

We're going to start sounding like a broken record in this e-book. It doesn't matter whether you have a simple, single-page application or a large, enterprise-scale web application, protecting your users and their data is paramount. Over the last few years, the use of JavaScript has ballooned. JavaScript, as described earlier, is flexible, easy to use, and easy to modify. Troves of developers prefer to use it for building unique and user-friendly web applications.

With increased use comes not only innovation, but also increased issues. Over the past few years, there are many processes previously done on the server side that have now moved to the client side. In some cases, this is a great development; in others it produces more challenges.

So, as Uncle Ben said to Peter Parker in Spiderman: "With great power comes great responsibility."

Authentication

One of the key technologies that still sits in both the client side and the back end is identity management. While you can roll your own authentication, it's more common to use a third party whose sole focus is providing secure identity services.

When the user goes through the authentication process, they are issued a token inside the browser which can be used for subsequent logins (until expiry). On the front end of the application, the developer can use this token to provide dynamic navigation, profile pages, and more.

Authorization

Typically controlled by the back end through middleware and role-based access, authorization has less of a play on the client side but is still important to understand for completeness of security when it comes to protecting web applications. Notably, broken access control is one of the biggest risks to web applications today, moving up to the number one spot in the OWASP Top 10.

JavaScript Code Testing

The best part about using JavaScript to develop web applications is that it allows for a lot of freedom and flexibility. Developers love to code sophisticated JavaScript applications, but they do come with significant risks to reliability, usability, and security. It is extremely important to test your JavaScript code during the software development lifecycle (SDLC) and throughout the lifetime of the use of the code. Testing during the SDLC can help you scale your website application faster and easier, while also ensuring code realiability.

There are a variety of automation testing technologies. **Allow us to walk you through the main three.**

Static Application Security Testing (SAST)

In Static Application Security Testing (SAST), the source code of an application is analyzed in order to evaluate if the code is secure or if pieces of the code can be replaced with a more secure code set.

SAST as a practice is easy to adopt, with the scanning process taking little time. Scan results can provide great feedback to improve your web application. This feedback can be incorporated directly into your integrated development environments (IDE) or can also be executed as part of your continuous integration/continuous delivery (CI/CD) pipeline.

Dynamic Application Security Testing (DAST)

In Dynamic Application Security Testing (DAST), software developers and security analysts evaluate web applications through the client side or front end, to find vulnerabilities by simulating attacks. By evaluating applications from the outside (i.e., from the perspective of the customer or threat actor), developers and analysts can find vulnerabilities and likely attack vectors. Once vulnerabilities and potential exploits are identified, developers can fix potential issues to ensure the code is secure.

Runtime Application Self Protection (RASP)

RASP is a technology that runs on a server and is designed to detect attacks on an application in real time. RASP can protect applications from malicious input or behavior by analyzing the app's behavior and the context of that behavior. By continuously monitoring the application's behavior, attacks can be identified and mitigated immediately without human intervention. This allows businesses to determine if they are under attack and quickly remediate any security issues.

How Do I Keep JavaScript Safe?

The best way to improve JavaScript security is through the use of scanning tools that detect, identify, and alert on behavior anomalies, and through the use of automated JavaScript-specific security policies that can automatically apply security configurations and permissions to help continuously monitor and protect malicious client-side activities.

Other things organizations can do to improve their overall JavaScript security include:

- **Use secure software development practices:** Apply best practices that enable the development of more secure application code and also aid in the detection and elimination of errors early in the application development process.
- **Move security to the left:** Security can't just happen after a web application is built or installed on a system. It needs to be a part of the entire website and application development process—from beginning to end.
- **Audit your web assets:** Know what web assets you own and the type of data they hold and regularly conduct deep-dive scans to reveal intrusions, behavioral anomalies, and unknown threats.
- **Maintain safe JavaScript libraries:** Confirm the security of any external libraries by making sure they're not on any blacklists. Regularly patch and update your libraries and avoid any dependence on third-party library sources.
- **Be selective with third-party scripts:** Third-party JavaScript is a great way to avoid the time and money associated with developing your own code, but third-party scripts can also contain vulnerabilities or intentional malicious content.
- **Use automated monitoring and inspection:** Monitoring and inspection activities are critical, but also time consuming if you don't have an automated solution to regularly review JavaScript code. A purpose-built solution that automates the process can be a fast and easy way to identify unauthorized script activity.
- **Validate input:** Cross-site scripting (XSS) risk can be minimized by validating input before invoking JavaScript functions.

04

JavaScript Security Approaches & Technologies

Protecting client-side web applications and websites is a goal that straddles both the application development and cybersecurity industries. Bugs and vulnerabilities in web applications represent a significant portion of the most common attack paths. However, there is some confusion about who is responsible for protecting the client side. Is it the security team or is it the application development team? While application security is shifting further to the left, both teams need to improve collaboration throughout the software development lifecycle to better integrate security earlier in the process and keep the client side safe from e-skimming, Magecart, formjacking, and other attacks.

To protect their customers from client-side attacks, businesses have seven primary tools at their disposal:

1. Web application firewalls (WAF)
2. Content security policy (CSP)
3. Penetration testing and assessments (vulnerability and security)
4. Client-side vulnerability scanning
5. Code scramblers and obfuscators
6. Client-side attack surface monitoring
7. JavaScript security permissions

Web Application Firewalls

A WAF helps businesses protect their web applications by filtering and monitoring HTTP traffic between the application and the internet. It protects web applications from attacks, such as cross-site forgery, cross-site scripting (XSS), and SQL injection. WAFs are deployed in front of web applications and analyze bidirectional web-based (HTTP) traffic, detecting and blocking anything malicious. They are great tools to use when protecting your business from skimming attacks, but they can only do so much.

What WAF Limitations Do I Need to Be Aware of?

WAFs are a special category of firewall that are designed specifically to protect web applications and your business from falling victim to skimming attacks. WAFs are deployed in front of web applications to analyze web traffic in order to detect and block malicious or unauthorized activity. According to the Payment Card Industry Data Security Standards (PCI-DSS), a WAF sits between a web application and the client endpoint and serves as a security policy enforcement point.

It is important to note, however, that WAFs are an open systems interconnection (OSI), layer 7 defense mechanism against application-layer attacks. They protect services that user-facing web applications apply to collect, store, and utilize data. WAFs are not designed to protect the browser-level user interface itself. In other words, if a web application and its user experience is a house, then the WAF protects the walls, not the furniture or the people inside. In the end, WAFs are not able to detect and protect businesses from sophisticated skimming malware, drive-by skimming, supply chain attacks, or side loading and chain loading attacks.

WAFs cannot protect businesses or their customers from:

- **Advanced Skimming Malware**—WAFs are not able to detect and protect businesses from more sophisticated skimming malware.
- **Drive-by Skimming and Supply Chain Attacks**—WAFs are unable to detect manipulated JavaScript code or data exfiltration.
- **Side Loading and Chain Loading Attacks**—WAFs do not protect against skimming performed by a sideloaded JavaScript code.

Are WAFs Right for Me?

Absolutely. WAFs are great security tools to help protect your business from client-side attacks, however, they can only block some client-side threats. They cannot block all of them. As with everything in security, there is no silver bullet to protect your business and your customers from all cyberthreats. A WAF will protect the connection between your servers and your customers, but the protection ends there. They can't monitor or protect your business from browser-level threats outside of your security perimeter.

Content Security Policy

A Content Security Policy (CSP) is an added layer of security that helps businesses and security teams detect and mitigate certain types of client-side attacks. A CSP can help uncover cross-site scripting (XSS), JavaScript code injection, and data skimming attacks.

What CSP Limitations Do I Need to Be Aware of?

First, it's not easy to add CSP to an existing website. Most websites and web apps are assembled using third- or fourth-party JavaScript libraries and code. Since web browsers load these scripts from external website domains or subdomains (e.g., code[.]company[.]com) developers end up whitelisting all external and internal hosts of scripts to avoid breaking the code's functionality.

What this means is that application development and security teams face a tradeoff between security and functionality of their websites. Whitelisting removes or reduces the very protection CSP is supposed to provide, thereby making the value of CSP inherently questionable. The complexity involved in building and maintaining CSP makes it easier for threat actors to find holes within those policies in order to steal data, distribute malware, or deface websites.

There are four main weaknesses in CSP that can expose you to e-skimming breaches:

- Excessive "allow list" rules
- Excessive whitelisting
- Incorrect CSP implementation
- CSP implementation tradeoffs

Is CSP Right for Me?

Sure, that is, if you have the time and energy to deploy and manage CSP on a continuous basis. CSP is a great way to launch a pseudo-Zero Trust approach to protecting your web assets, but, as described earlier, CSP comes with significant risks. CSP is also extremely vulnerable to cross-site scripting attacks. For example, even when CSP is applied, scriptless or post-XSS attacks can still be executed, some XSS vulnerabilities simply aren't mitigated, and some browsers do not support CSP at all. So, while CSP can add a level of increased client-side protection, it will only stop a limited number of client-side threats.

Client-side Pentesting, Vulnerability, and Security Assessments

Penetration Testing

A penetration test, more commonly referred to as a "pentest," is a deliberate cybersecurity attack, conducted with permission from the organization by professional cybersecurity experts and designed to uncover weaknesses and vulnerabilities across an organization's security controls. Companies either use internal red teams to carry out these attacks or hire an external security service provider that specializes in penetration testing. During the pentest, red teams attempt to enumerate and infiltrate their target's digital infrastructure, networks, and endpoints. Once vulnerabilities have been identified, pentesters try to mimic threat actor tactics, techniques, and procedures (TTPs) to forage deeper into their target's systems and networks. The final output of the pentest is a report that outlines what security gaps exist and what needs to be addressed to secure the business from cyber threats.

Vulnerability Assessments

A vulnerability assessment is a systematic analysis and review of security weaknesses in a technology, system, application, or network. During these assessments a security analyst will determine if the system is susceptible to any known or exploitable vulnerabilities, assign severity levels to them, recommend remediation or mitigation, and prioritize the order in which remediation must occur based on the severity level.

Security Assessments

Unlike pentesting and vulnerability assessments, which are focused on the tools and technology, security assessments are more concerned with process, governance, and compliance. Essentially, it is an evaluation of your tools, applications, websites, and technologies, and how they are used to determine if they are secure from cyber risks and threats. Security assessments identify and evaluate if your business has the proper security policies and controls in place across all of your assets inside and outside of your security perimeter. The end result of a security assessment should be deep insights into the security gaps of your organization, aligned to both your overall security program and a governance model (e.g., NIST). The undertones of the report should also provide a risk level of your organization in its current state. Generally speaking, security assessments are a core piece of any organization's risk management process.

Security assessments can be performed by consultants or internal teams. It depends on how mature the organization's processes and security teams are. Consultants and security analysts who perform assessments conduct in-depth audits of the organization's defensive measures against various attack methods used by threat actors, including internal staff (insider threat and human error) or external actors (hackers trying to breach the business via client-side or server-side attacks).

What Is Tested During a Client-Side Security Assessment?

Client-side security assessments are actually quite uncommon at this point in time. Unfortunately, this lack of client-side assessments presents a huge problem given the dramatic rise in client-side attacks like cross-site scripting, formjacking, and Magecart. With the increased use of front-end frameworks, libraries, and third-party tools, it's time for organizations to expand the scope of traditional security assessments and testing to include the client-side attack surface of their websites and web applications.

Client-side security assessments are tedious if done manually. **There are five categories of questions a consultant or security analyst needs to answer to uncover potential client-side issues and their associated risks:**

01

What Client-Side Assets Do We Have?

If you don't know what you have, you can't protect it. The first step in a security assessment is to inventory all webpages, web applications, landing pages, forms, payment forms, marketing trackers, and other client-side assets that might pose a risk to the business if corrupted.

02

What Technologies Do We Use? What First- and Third-party Code Are We Using? What Does Our JavaScript Supply Chain Look Like?

This is critical. Websites today are assembled in real time using a variety of protocols, connections, and data sources. Businesses must have an inventory of all webpage and web application components. Assessors need to have a full picture of all their scripts, where they are loaded, how they are loaded, and how they interact with other JavaScript code across the client side. The easiest way for threat actors to steal protected data is by corrupting third-party JavaScript. Without continuous client-side testing, you might never know a threat actor has breached your JavaScript supply chain and is stealing customer information.

03

Who Has Access to Our Data (in Real Time)?

Once you have a list of your assets and the associated technologies, it's time to start looking into who has access to them and what type of access they have. Are third parties reading all of your customer data during every form submission? How are you protecting our user's privacy? Being able to shape data access insights across the client side is the next big step in protection.

04 Are We in the Midst of an Attack Right Now?

Once you have a full inventory of your client-side pages, applications, and the code you use, it's time to see if your client-side web assets are only doing what you want them to be doing... ahem...that is, is the data you are collecting only being collected by you or is it being sent to a threat actor's command and control domain in Uzbekistan? You want to look at your keyloggers, your WebSockets, any anomalous behaviors, and if there is any data transfer to unauthorized countries or servers.

05 What Needs to Be Fixed Now?

Once the security assessor has inventoried your client-side assets and the code used to build and maintain them, and any potential breaches and exploited vulnerabilities have been uncovered, the assessor should provide a detailed report on what the organization's security team should do to secure the business. Client-side security assessments should point out:

Security configuration gaps:

- **Current access controls:** This identifies who currently has access to what and how to limit access to ensure only authorized individuals can modify or utilize client-side assets.
- **Overly permissive access:** Clear recommendations on how to deploy a Zero Trust approach to client-side web applications and websites to reduce the risk of tampering. This helps ensure who has full access, read only access, and data transfer access.

Malicious elements:

- **Malicious host scripts:** Are any malicious hosts actively stealing data? What can be done to fix this issue?
- **Malicious scripts:** Is the business currently using any first- or third-party script that has been corrupted and is exfiltrating data or modifying the webpage or application in any way? What can be done to fix this issue?

Vulnerabilities:

- **Exploited vulnerabilities:** Are there any known vulnerabilities currently being exploited? Is there a patch available to fix these vulnerabilities and which ones are the most critical to patch?
- **Other vulnerabilities:** Are there any known vulnerabilities that we can patch proactively to reduce client-side cyber risk? Is a patch available and how critical is it to patch the vulnerability now?

What Pentesting, Vulnerability Assessment, and Security Assessment Limitations Exist?

Typically, pentests, vulnerability assessments, and security assessments are performed as short-term projects that are repeated on a quarterly or annual basis. Finding good pentesters is hard and they demand a high wage because of the specialized skill set and experience they possess. Many organizations hire a managed security service provider (MSSP) to conduct the pentest.

Let's assume that a penetration test or assessment is 100% accurate and provides actionable results. That's great. However, results are a snapshot in time, which means hackers have the ability to execute attacks between quarterly or annual assessments. Additionally, hackers are always looking for new vulnerabilities to exploit and likely will know about new exploits before a pentest has been completed. Relying on quarterly or annual vulnerability assessments is a great start, but companies still remain exposed to breaches. Ultimately, threats and threat actors can move much faster than any company.

Penetration tests and assessments also present limitations because they:

- Are time and resource intensive.
- Are limited in scope to certain applications, technologies, and networks.
- Require a skilled consultant, tester, or employee with the know-how to be successful.
- Rely on the use of specialized tools and technologies to uncover vulnerabilities and threats.

Are Pentests, Vulnerability Assessments, and Security Assessments Right for Me?

Yes! Yes they are! They are a necessary aspect of any cybersecurity program. But keep in mind that they are not continuous. The information gleaned during a pentest or assessment represents only those issues that exist at that moment, and there might be a laundry list of new vulnerabilities and issues that a different pentest or assessment will uncover in a week or a month. Threat actors move faster than any government or business. To stay ahead of the threat, you need more than a period-in-time pentest or vulnerability assessment.

Client-Side Vulnerability Scanning

Software Vulnerabilities

Simply put, a software vulnerability is an error or defect in software. Threat actors and hackers love to scan networks, systems, applications, and software to find vulnerabilities that could be leveraged to gain control of the system or software. Vulnerabilities stem from the way the software is designed. For example, there could be a flaw in the way the application or software was coded, errors could have appeared in a software update or release unexpectedly, or code errors could have been injected inadvertently when first- or third-party code was added to the software or application.

Why Is JavaScript Vulnerable?

As discussed earlier in this e-book, JavaScript, like many other coding languages, wasn't designed with security in mind. Being able to protect JavaScript is something that we are now having to reckon with given the explosion of usage in all modern digital applications.

JavaScript vulnerabilities have become a favorite for threat actors given the ease with which they can be exploited and leveraged for more advanced attacks like digital skimming.

What Is Vulnerability Management?

Vulnerability management is the continuous process of identifying, analyzing, prioritizing, and remediating weak points in an organization's cybersecurity posture. Vulnerabilities include potential exposures or risks that need to be mitigated to ensure threat actors cannot exploit them to access the network for malicious purposes.

What Can Vulnerability Scanners Do?

There are quite a few vulnerability scanning products on the market today, many of which have been around for a long time. These tools are designed to scan and assess computers, software, applications, servers, and networks to uncover known weaknesses (a.k.a. vulnerabilities) that could be used for malicious purposes by hackers. Vulnerability scanners are used to identify and detect vulnerabilities arising from misconfigurations or flawed programming within network-based assets such as firewalls, routers, web servers, application servers, and more.

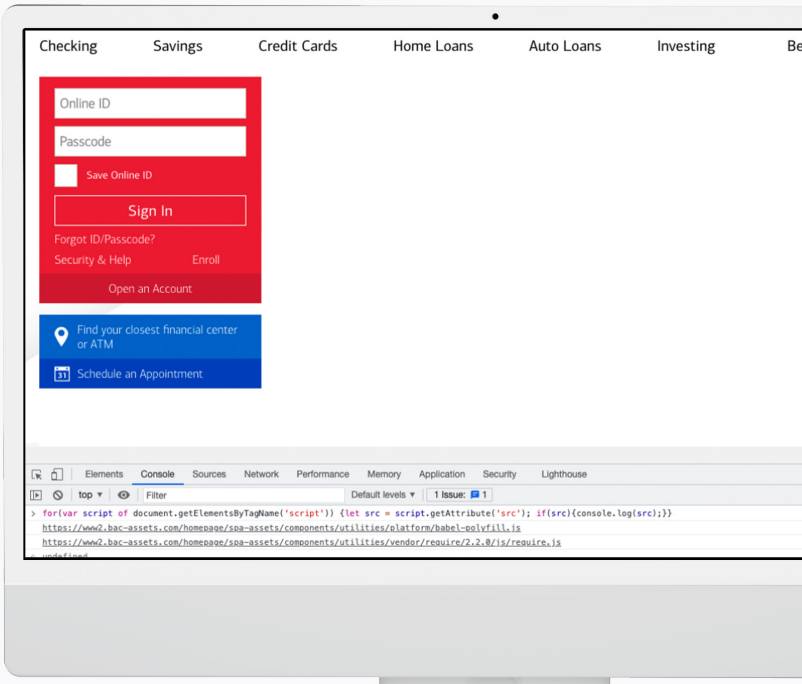
Cybersecurity teams deploy vulnerability scanners to find potential inroads hackers could use to breach their networks and defenses. Once a vulnerability has been found, vulnerability management and security teams then patch the vulnerabilities with software updates. If vendor software updates are not available, security teams find ways to reduce the potential harm or cyber risk they might incur if a hacker attempts to breach their network using the vulnerability as their entrypoint.

What Limitations Do Vulnerability Scanners Have When Used for Assessing Client-Side JavaScript?

Vulnerability scanners are designed to scan back-end code and systems, typically those digital assets that live on the server side. They aren't able to detect and enumerate all JavaScript scripts and vulnerabilities. To be blunt, they just can't see them. Vulnerability scanners can only see the client-side after it's been assembled together, not in real time. Vulnerability scanners also can only see a single domain, not all of the links that are part of it.

What Are Typical Vulnerability Scanners Able to See and Scan?

Traditional vulnerability scanners are more closely related to bots than humans. As such, there are several human-like actions they cannot replicate, which means they cannot detect client-side threats in their entirety. In this example, only a few JavaScript items are detected. The screenshot below shows this in a very simple way. A vulnerability scanner will only pick up a handful of active scripts.



Client-side security technologies are able to pick up

50+

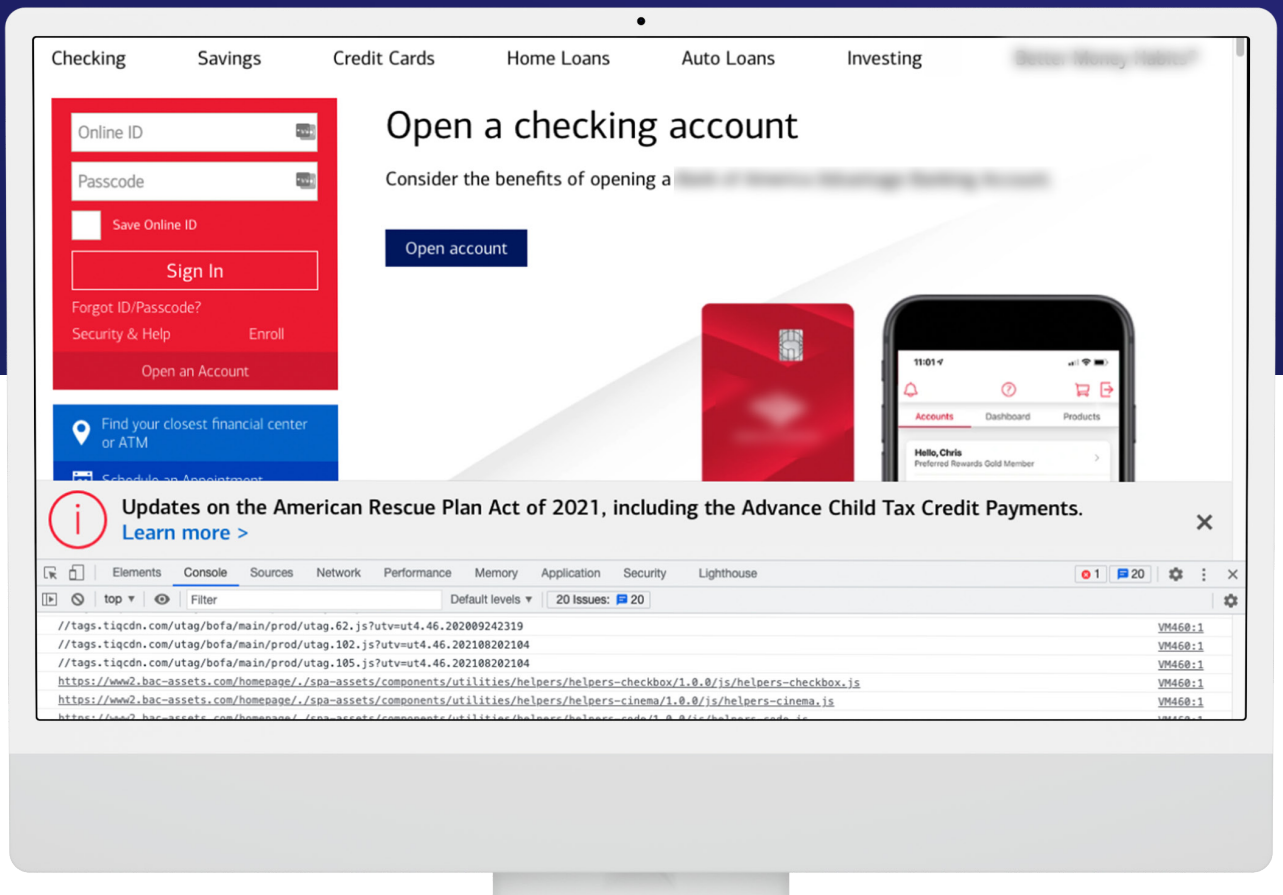
active scripts.

What Are Client-Side Security Technologies Able to See and Scan?

Client-side security technologies replicate actual user behavior on a webpage, including the ability to execute custom user journey scenarios. Similar to the image below, client-side security technologies are able to pick up more than 50 active scripts. Needless to say, there is a massive detection gap that leaves businesses vulnerable to client-side cyberattacks.

Are Vulnerability Scanners Useful for Client-Side Security?

Vulnerability scanners are a necessary technology of any cybersecurity program. However, they are not useful for client-side security. Vulnerability scanners are not designed to support client-side security efforts. There are client-side security technologies specifically built to scan JavaScript web applications and websites. They see all scripts, network requests, and resources in each scan. Since JavaScript is so easy to manipulate, threat actors can move very fast when hijacking them for malicious purposes. This type of technology continuously scans client-side web applications and websites, and alerts on issues immediately. Furthermore, a new product category recently hit the market which deploys JavaScript security permissions, so that hackers can't exfiltrate data. Security teams can rest easy that their client-side applications are protected, and they can fix vulnerabilities and issues when they are ready, not under duress.



Code Scramblers and Obfuscation Technologies

What are Code Scramblers and Obfuscators?

Code scrambling or code obfuscation is a process by which easy-to-read code is distorted to make it difficult to comprehend. The goal of code scrambling and obfuscation is to make it difficult for competitors, other developers, and threat actors to reverse engineer or modify it. Code obfuscators allow websites to load properly in the browser without being different to the naked eye. Web application developers and security teams acquire code obfuscators in order to hide JavaScript web application code that threat actors or competitors might target. By concealing portions of the code, developers hope to reduce the risk that threat actors might use the code for malicious purposes.

What Limitations Do Code Scramblers and Obfuscators Have?

It is quite easy and cheap to scramble or obfuscate code. There are many free code obfuscation tools available, but they come with some big limitations. For example, with time and effort, web applications that were obfuscated with free code obfuscating technologies can be reverse engineered to uncover a version of the original application. This is why hackers use code obfuscation to their advantage.

There are also code de-obfuscators on the market. These simply do not work. Some paid code obfuscating technologies pollute the original web application code to such an extent that you can't even get remotely close to the original code if you try to unscramble it. Herein lies a substantial problem. If you obfuscate your web application code, you can't unscramble it. You can't go back to the original code. So it gets extremely hard to spot security issues and vulnerabilities in the code. Normal code and malicious code in a scrambled web app looks exactly the same. Web developers can no longer see issues in the code with the naked eye and are likely to miss critical vulnerabilities. Some businesses circumvent this by implementing dynamic code analysis to track issues, but this is time consuming, costly, and inefficient.

Are Code Scramblers and Obfuscators Right for Me?

Well, it depends. If you have a simple, single-page web application that doesn't collect user data and security isn't an issue, then, sure, code obfuscation might be a viable option for you. However, if you have a sophisticated web application or webpage that you use to interact with your customers, you cannot risk not being able to see vulnerabilities in your code. If your web application is using third-party scripts, and those are attacked in a drive-by web skimming attack, you will not be able to see the malicious code. A skimming attack could run on your web application in perpetuity. If your web application collects payment information, or if you conduct business in Europe where GDPR violations are a clear and present danger, you are leaving your business open to tremendous fines, legal action, and troves of upset customers

Feroot Security
Inspector

Client-Side Attack Surface Monitoring

What Are Client-Side Attack Surface Monitoring Solutions?

Client-side attack surface monitoring solutions are a relatively new cybersecurity technology that automatically discover all of a company's web assets and reports on their data access. These solutions use headless browsers to navigate through all the JavaScript contained on the website and web application pages. The technology gathers real-time information about how the scanned website works from the end-user perspective.

A key component of the technology is synthetic users, which are deployed during threat detection scans to act and interact the way a real human would when completing websites and web application tasks. These synthetic users can complete a variety of activities, including but not limited to:

- Scrolling through pages
- Submitting forms
- Solving CAPTCHAs
- Entering financial information
- Clicking active links
- Watching embedded videos
- Waiting for pages to load
- Navigating between pages
- Clicking on, opening, and closing pop-up messages

Each interaction conducted by a synthetic user with the web application is logged and monitored. These solutions then engage behavioral analyses and inject logic into each page to gather information that is difficult to collect manually, including:

- The type of data collected by forms.
- The type of data third-party scripts have access to.
- Any first- and third-party scripts that are fingerprinting users and their browsers.
- The types of trackers that are deployed on the page and their activities.
- The existence of any forms or third-party scripts transferring data across international boundaries or to unauthorized entities.
- Any first- and third-party scripts which are being loaded directly into the user's browser.
- Any first- and third-party scripts that are being sideloaded or chainloaded into the user's browser.
- The presence of any malicious hosts exfiltrating data.
- Whether any data is being exfiltrated via websockets and the location where it is being exfiltrated.

Evaluation of web applications from a security perspective isn't the only thing that client-side attack surface monitoring solutions do. They also perform post-scan informational analyses to offer businesses synthesized intelligence to secure web applications from harm.

In addition, they analyze all information synthetic users collect and enumerate client-side threat intelligence for security teams to act on quickly and effectively. Built-in machine learning capabilities also identify and classify data to detect and report on a variety of client-side security challenges. The type of intelligence available includes:

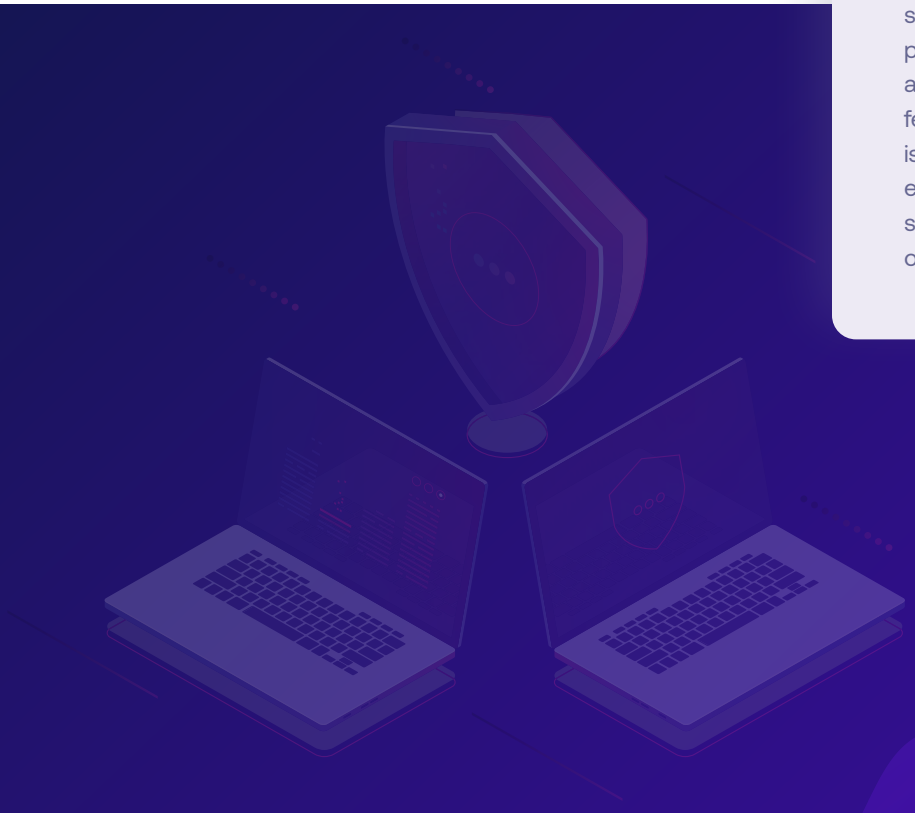
- Active malware
- Live marketing or other tracking software
- Geographic IP information
- Obfuscated scripts
- Data assets collected (financial, PII, etc.)
- Historical overview of your client-side attack surface
- Client-side security trends
- Types of webpages (login, billing, etc.)
- SSL issues
- Known JavaScript vulnerabilities

What Limitations Do I Need to Be Aware of?

The limitations are minimal for this client-side security approach. Client-side attack surface monitoring solutions are easy to set up and maintain on existing web applications and can discover more client-side cyber threats than any of the approaches discussed in this e-book. These cyber defense technologies do require interaction between cybersecurity and application development teams. To properly secure businesses from client-side threats, teams need to understand the ins and outs of client-side application structures, as outlined earlier in this e-book. With strong collaboration between security and application development teams, businesses can secure their client-side web applications with ease.

Are Client-Side Attack Surface Monitoring Solutions Right for Me?

If your business interacts with customers via web applications or webpages, then yes, client-side attack surface monitoring solutions will enable your business to stay ahead of client-side cyber threats. Client-side attack surface monitoring solutions condense manual processes that typically take security analysts and web application developers days into just a few minutes. With automated alerts and detailed issue enumeration, these technologies can enable security teams to automate client-side security tasks beyond any scope available with other client-side security approaches.



Feroot Security
PageGuard

Client-Side JavaScript Security Permissions

I know what you're thinking. There is no such thing as JavaScript security permissions!!! Hold up! There is a new product category to address the growing client-side security gap.

What Are Security Permissions?

Permissions are a way for application developers and security analysts to control access to a specific system and device-level functions in an application, page, or other software. Traditional applications, that is, those not written in JavaScript, generally come with a menu of options or functions that may be made visible or hidden from a user based on their permission level. Most permissions must be granted at runtime by the user. The user has the right to revoke permissions at any time.

Types of permissions include the applications ability to:

- Access features on the user's machine, such as their camera or mic.
- Review and collect personal data, such as personally identifiable information, data entered into forms, IP address, and location data.
- Grant rights to modify the functionality of the application or software.

What Are JavaScript Security Permissions?

Traditional software and applications, a.k.a. those not written in JavaScript, come with a menu or functions to set user permissions. In contrast, JavaScript is the wild, wild west. By default, JavaScript environments do not have a security permissions model built in. Third-party JavaScript code can have an unrestricted level of access to sensitive data at the browser level, so the attack surface is broad and wide open. So do JavaScript security permissions exist? Yes, they do.

JavaScript security permissioning technologies add security permissions and controls to JavaScript. Application developers and security teams simply have to add a few lines of code to their websites and web applications, then these solutions automatically apply security configurations and permissions for continuous protection from malicious client-side activities and third-party scripts. These products integrate directly into the runtime environment of every user browser session to enable proactive monitoring and defense.

They essentially deploy a Zero Trust model on JavaScript applications and run continuously in the background to automatically detect unauthorized scripts and anomalous code behavior. After detection, they block all unauthorized and unwanted behavior in real time across an organization's web applications.

In short, a JavaScript security permissioning solution monitors and responds to browser-level security events in real time by auto-instrumenting itself on every website and by applying security configurations to every user browser session. I assure you, it's not too good to be true.

What JavaScript Security Permission Limitations Do I Need to Be Aware of?

There are none. If an application development or security team deploys JavaScript security permissions on all of their client-side pages and applications, then third-party JavaScript code can't be tampered with and data can't be exfiltrated by threat actors. Coupled with proactive scanning of client-side assets, application security and cybersecurity teams will receive alerts with context, to repair client-side security issues, all while being protected.

Are JavaScript Security Permissions Right for Me?

If you work for a company that uses marketing landing pages, e-commerce technologies, user portals, and other technologies to communicate and collaborate directly with your customers, then **YES, you do need to add JavaScript security permissions!**

05

JavaScript Risks & Threats

There are dozens of methods that cyber threat actors can use to steal personally identifiable information (PII), financial transaction data, and other confidential and proprietary data from businesses. Traditionally, this involved breaching a target's corporate network. Today, there are a multitude of new vectors that can be used to steal data from companies, including the use of malicious scripts directly on the front end of a website or web application to start skimming data as a user enters information into a form. Digital skimming is a client-side cyberthreat that requires improved security on any website or web application to protect customers interacting with the website from losing sensitive information.

JavaScript Security Threats and Attacks

Even as businesses are shoring up JS vulnerabilities, bad actors are stepping up their threats and attacks. Fortunately, the most popular of these attacks are well known. By educating your staff and organization on the most common client-side security threats, it's easier to learn how to stop them in their tracks.

01 Client-Side Vulnerability Attacks

Cross-Site Scripting

Cross-site scripting (XSS) is one of the most common ways for bad actors to launch a client-side attack. In a typical XSS attack, malicious code is injected into website content which targets unsuspecting users viewing that site. This threat is so common that it's regularly on The Open Web Application Security Project's (OWASP) list of Top 10 Vulnerabilities.

DOM-Based XSS

In a document object model (DOM)-based, cross-site scripting (XSS) attack (sometimes called "type-0 XSS"), the threat actor's payload is executed as a result of modifications to the DOM environment in the victim's browser, which was used by the original client-side script. As a result, the client-side code runs differently than originally designed. The page itself doesn't change, but the client-side code on the page executes malicious DOM-environment modifications.

Directory Traversal or Path Traversal

Directory traversal or path traversal attacks exploit weak security validation or sanitization of user-supplied file names in order to gain access to files and directories that are stored outside of the web root folder. In a path traversal attack, threat actors manipulate the variables that reference files.

02 Web Skimming Attacks

E-skimming

E-skimming, commonly referred to as a 'Magecart' attack, is a process in which malicious threat actors, nation-state-sponsored hackers, and financially motivated hackers gain access to e-commerce sites. These threat actors inject skimming code onto payment card processing pages of the website in order to make financial gain. E-skimming is also increasingly targeting third-party vendors who provide web analytics or online advertisements.

Magecart

Magecart is a commonly used name for threat groups that use digital skimming or e-skimming techniques to steal customer data. The term "Magecart" is a portmanteau of the words "Magento" (a popular open-source ecommerce software platform) and "shopping cart." It operates using a small piece of code, which is inserted into a website to skim information from customers. Most often, it's inserted into payment pages, where it acts as an online credit-card skimmer, pulling the personal information and credit card details of anyone who comes across it.

E-commerce Platform Skimming

Beyond Magecart, some of the world's most popular e-commerce platforms, like Volusion, have also been breached by Magecart e-skimming code. These e-commerce platforms provide checkout services to thousands of merchants, while collecting and utilizing massive amounts of customer credentials, personal data, and financial information. Thousands of online stores were compromised during the Volusion platform hack in 2019, which went undetected for weeks.

Drive-by Web Skimming

In drive-by web skimming, a threat actor compromises third- or fourth-party code with malware, with the hope that multiple organizations use this code and infect their websites and web applications inadvertently. Modern web applications load an average of over 20 third- and fourth-party scripts as part of the user experience. Compromising one of these third- or fourth-party elements with malicious JavaScript allows an attacker to compromise multiple websites simultaneously.

Trusted Cloud-Hosted Platform Skimming

Malicious e-skimming code has been found hosted by popular cloud platforms, including Salesforce, Heroku, Amazon CloudFront CDN, and many others. Magecart e-skimmers were found following a spray and pray approach that targeted misconfigured Amazon S3 buckets. Skimmers were able to open backdoors and insert malicious code into JavaScript libraries used by thousands of organizations.

Public Wi-fi Skimming

IBM researchers discovered this type of e-skimming attack on public Wi-Fi hotspots. In a public Wi-Fi skimming attack, threat actors infiltrate a large number of users in public spaces, such as airports and hotels by skimming off unprotected public Wi-Fi. Threat actors insert skimming code via Wi-Fi hotspots allowing them to exfiltrate the data users enter on web forms. These forms may include e-commerce checkout pages, marketing landing pages, or login pages. Public Wi-Fi skimming is quite a simple attack for threat actors to execute, since vulnerable Wi-Fi routers allow attackers to automatically inject skimming scripts into all websites accessed by users through their connected devices.

Anti-forensic, Self-Cleaning, and Stealth Data Skimming

There are a few variants of web skimming codes that are notoriously difficult to detect and remediate. Pipka is a web skimming code with anti-forensic, self-cleaning, and stealth capabilities. It is able to remove itself from a webpage's code after it has been executed, thereby making it exceptionally difficult to detect. Pipka-like threats require focused attention by cyber defenders and an automated scanning solution that alerts them to questionable code behaviors.

03 Malicious Script Injection Attacks

JavaScript Injection

During a JavaScript injection attack, a hacker gains website or web application parameter information and changes their values. This allows the threat actor to manipulate the website or application and collect sensitive data, such as PII or payment information. Threat actors tend to use JavaScript code obfuscators or scramblers to make it difficult for web application developers or security experts to see the malicious code and detect threats.

XML Entity Injection

XML External Entity (XXE) Injection is a web security vulnerability that allows attackers to interfere with an application's processing of XML data. Attackers exploit these vulnerabilities to view files on the application server file system and to interact with any systems that the application can access.

Formjacking

In formjacking, cyber criminals, financially motivated threat actors, and other hackers insert malicious JavaScript code into their targets' website. Their goal is to take over the functionality of the site's form pages to collect sensitive user information or valuable data. This information can include credit card information, personally identifiable information (PII), and other data that can be used to breach networks or be sold on the dark web.

Sideloading and Chainloading

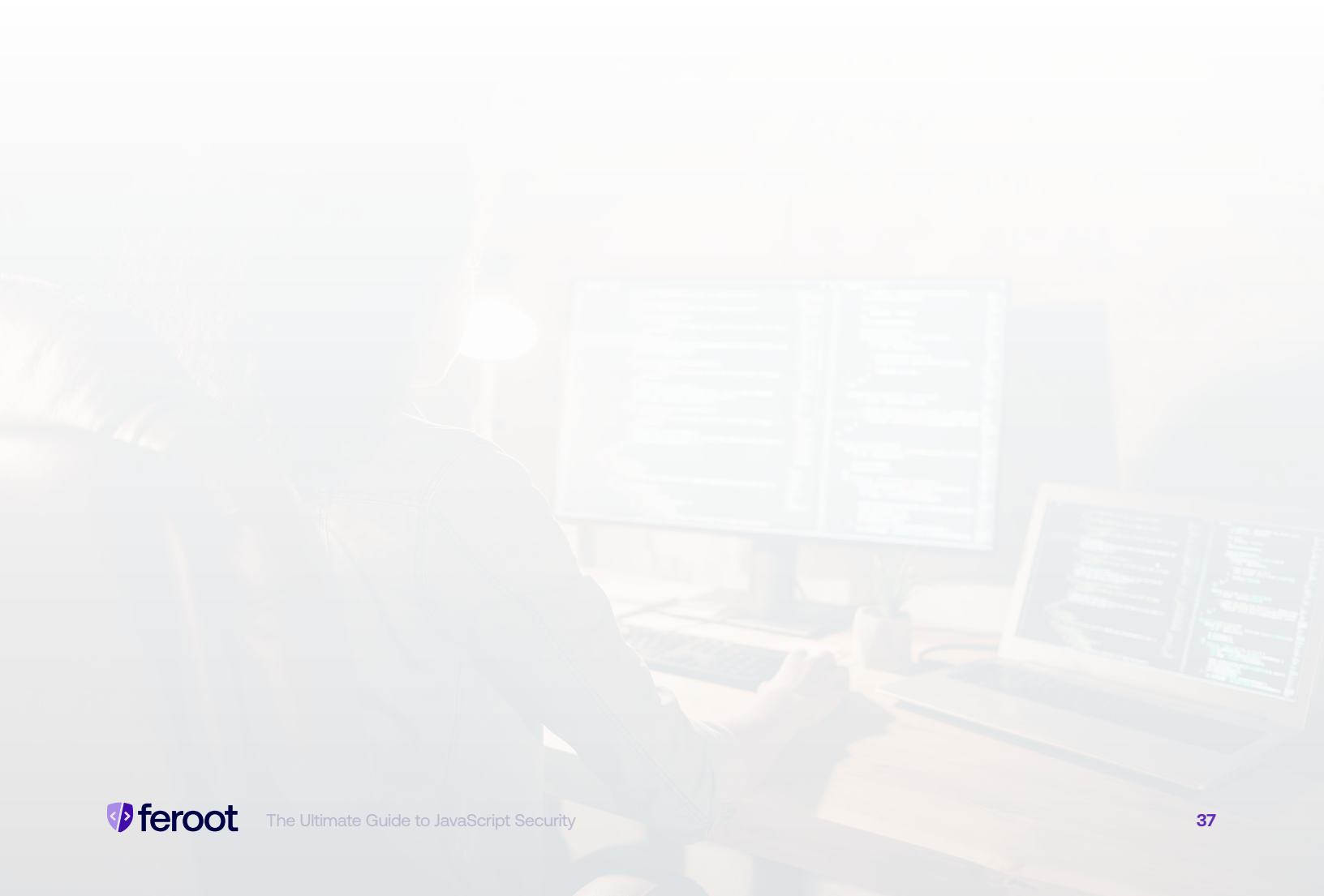
Side loading and chain loading are threat actor attack techniques that allow them to load malicious or infected JavaScript code onto a target webpage using legitimate scripts and tools.

JavaScript Sniffing

A JavaScript sniffer is a form of malware that is designed to steal financial transaction data from websites, web applications, and online forms when a customer decides to purchase a good or service through an e-commerce website or e-store.

Broken Link Hijacking

Companies that are not careful and leave unused, expired, or invalid links on their websites or web applications put themselves at risk for broken link hijacking. This client-side attack relies on companies forgetting about broken links, which are then hijacked and used to load malware or other types of malicious code into the user's browser. To protect your company from broken link hijacking, ensure that all links that are no longer in use are removed or replaced immediately.



04 Request Forgery

Server-Side Request Forgery

Server-side Request Forgery (SSRF) is an exploit where a threat actor abuses the functionality of a server causing it to access or manipulate information that would otherwise not be accessible to the hacker. One example of a SSRF attack includes a hacker causing the server to connect to services within the organization's infrastructure. Hackers could also force the server to connect to external systems to exfiltrate data such as authorization credentials like usernames and passwords.

Cross-Site Request Forgery

Cross-Site Request Forgery (CSRF), also known as a one-click attack or session riding, is a type of cyberattack that forces an end user to execute unwanted actions on a web application in which they are currently authenticated. CSRF directly manipulates end-user browsers.

JavaScript Attacks and Threats: Who Is at Risk?

Threat actors love to exploit the obvious, so perhaps a better question is "who is not at risk?" To do business with any consumer, organizations must have a digital presence. Marketers like to say "our website is the center of our universe," and this definitely holds true. If a business does not have a website, consumers quickly question the validity of the business. Buying behaviors today dictate that a business must have an online presence available for the consumer to evaluate the goods and services sold as the first step in the consumers' decision-making process.

That being said, threat actors also don't like to waste their time or money. They tend to go after low-hanging fruit in order to maximize their return on investment, especially if they purchased malware on the dark web and need to recoup that initial investment. Threat actors also try to attack many targets at once in order to hedge their bets. All they need to recoup their initial investment is to breach one website.

Industries at Risk

There are a number of industries at risk of e-skimming attacks. Below is a sample of the industries and their estimated e-skimming risk level.

High Risk

- Financial Services & Banking
- Insurance
- Healthcare & Medical
- E-commerce & Retail
- Travel & Hospitality
- Communication, Social Media, & Content Producers
- Cryptocurrency Exchanges & Blockchain

Medium Risk

- Real Estate
- Technology & Cybersecurity
- Distribution & Transportation
- Education
- Entertainment

Low Risk

- Manufacturing
- Energy
- Distribution & Transportation
- Consulting & Legal Services

06

JavaScript Security Teams and Collaboration

Protecting JavaScript web applications is a team sport. It doesn't matter if you are building the next awesome web app or webpage or are detecting and defending your business from threats, you will need a great team that has your back to ensure your JavaScript web apps are safe and secure for your customers to use. To successfully secure JavaScript applications, there are four core teams which application development professionals need to collaborate closely with to avoid costly Magecart-like attacks or other client-side breaches.

Cybersecurity Teams

Web application developers must build strong relationships with their cybersecurity teams in order to keep web applications secure. Understanding how cybersecurity professionals keep the business safer from cyberthreats will help with testing and security of web applications in production and test environments.

Goal Alignment

First things first, web application developers are constantly under pressure to deliver new features, fix bugs, and innovate faster...all of which goes against the grain of the cybersecurity professional's role in the business, which is to make sure the web application and the code it is built on is as secure as possible.

Once a web application has reached the point where it can be sent to a security team for testing, the fun begins. The security team will poke holes into the application to ensure that it will not harm the business. While this can be painful for application developers, it is critical that they go back in and fix vulnerable code. A development sprint schedule might have to be adjusted, which can delay release dates of new apps, features, or functionalities. However, repairing vulnerabilities will reduce the company's cyber risk. What this means is that the company will have a lower chance of being breached, the cost of which can be millions of dollars.

Web application developers and security professionals are quite similar in nature. Both possess deep technical understanding and experience. There is a massive amount of potential to build and maintain strong partnerships across the two groups and it all is based on mutual understanding.

Keeping a business safe from cyberthreats is a tremendous undertaking. Security practitioners work nights, weekends, and holidays to keep their businesses and colleagues safe. Threat actors work odd hours and attack businesses from international locations, thereby ensuring that security practitioners have to be on their toes constantly. Security practitioners live in a high-stress world in which they never know when the next vulnerability will be exploited, when the next zero-day malware will breach their network, or when John in accounting will click on another phishing email. Security practitioners try their best to eliminate as many threats as possible as quickly as possible. What might seem like a non-issue to a developer could snowball into a large-scale breach if threat actors are able to move laterally through different applications.

Let's be honest. Some developers do not have the security acumen to understand what a specific vulnerability means or how to fix it. Work closely with your cybersecurity team to learn from them. Security experts will be patient and will collaboratively walk through vulnerabilities and potential solutions to fix them with application development teams. If there is a knowledge gap, do not hesitate to ask the cybersecurity team for training on security concepts, vulnerability management, and what the outcome of a vulnerable application might be. They will be more than happy to show you the ropes. Get to know your security team and work with them as fast as possible to fix issues. Keep in mind that they likely will have to work late to keep the business safe. Anything you do to help the security team during normal business hours will make everyone's life easier in the long run.

Marketing Teams

Marketing and application development teams are getting more closely aligned by the minute. Marketing professionals own the customer experience, which is driven by websites and web applications. Just like web application developers, marketing wants to get cool stuff out, have it be easily accessible, and ensure prospects and customers are engaged when they interact with the business. Marketing professionals are there to learn how customers engage with your web applications. They are charged with showing you what the user experience is, so that you can enhance existing or develop new cutting edge web applications that delight your users.

Goal Alignment

To those who are unfamiliar with marketing, a lot of what marketers do can seem strange, intimidating, or just plain annoying. It is the marketing team's job to make sure prospects know the business exists and to convince them to interact with the business by reading blogs, downloading content, and sharing their contact information. Collecting a prospect's contact information and safely storing that information is key to building business relationships, conveying product/service value propositions, and, ultimately, generating revenue. To a security practitioner, this might open up additional avenues for cyberattacks, but it is a necessary condition to run and grow a successful modern business. Afterall, this isn't the 1980's and a yellow pages listing no longer works to drum up business.

Reducing website breach risk is crucial because website attacks are a very real and dangerous thing. Fortunately, marketing and application development teams can have a fantastic partnership to drive business growth. Application developers need to keep in mind that most marketers are not technical people. They likely have never coded anything in their life and most of the technologies and tools developers use are quite foreign to them. So, to communicate effectively with your marketing team, explain why you built your web application in the way that you did, what its strengths and benefits are, and what limitations you might have. Marketers might ask for things you can't deliver, but they will listen and learn and roll with your recommendations. Hey, their crazy ideas might actually help you build something new and innovative that you can post to GitHub for others to use.

Let's be clear here, the majority of your marketing team is not able to build webpages or web applications. They are not developers and likely can't tell you what JavaScript is. They are just trying to make your business look good to generate demand for your services. Marketers have a completely different, but equally valuable, skill set. Keep this in mind when you collaborate with the marketing team.

Good developer teams and marketing teams work together to promote the business together, remove any friction in the customer journey, and enable each other to be successful. If you are a web application developer, work with the marketing team to learn about the customer journey so that you can figure out how you can enhance it. Marketers help your business grow so that you get a paycheck or perhaps get some new colleagues to reduce your workload. Even better, marketers are great at generating ideas for new and innovative projects that can further your career. Get to know them and teach them what they need to know about website and web application development. Marketers tend to have open minds and love learning from experts. To a marketer, knowledge is power. Share your knowledge with them and they will support you through thick and thin.

Privacy and Compliance Teams

The European Union imposed the General Data Protection Regulation (GDPR) regulation a few years ago. Ever since then web application developers, cybersecurity professionals and legal teams across the globe have been feeling a tremendous amount of pressure to protect customer data and ensure adherence to data privacy regulations. Every business collects personally identifiable information (PII) on their customers. Depending on the industry, you can add additional types of PII to the security roster, such as personal health information (PHI).

Goal Alignment

To a privacy and compliance professional much of what web application developers do is magic. Their education and professional day-to-day job duties aren't anywhere close to a developers. Privacy and compliance professionals want to make sure customer data or proprietary information isn't stolen, sold, or leaked. They will likely rely on the cybersecurity team to work with the web application development team to find the right tools, processes and technologies to effectively secure corporate and customer data. They might even throw some budget your way to ensure privacy and compliance projects, tasks, and ongoing processes are completed successfully.

Privacy and compliance teams will want to see reports on what has been done to protect user privacy and what protective measures have been put in place to secure the business from the privacy perspective. Before you take on a new privacy and data security project, talk to your privacy and compliance team and your cybersecurity team to outline goals and set expectations. Let the cybersecurity team lead the charge to get buy-in on outputs and outcomes from the privacy and compliance team early. Get your ducks in a row to avoid rework and to meet everyone's expectations. Also, don't get frustrated with your privacy and compliance if a new regulation pops up and you have to change code or deploy additional security measures. After all, they have no control over governments and the policies that are enacted.

Product Security Teams

Cyber threat actors have pushed businesses to not only invest mountains of resources into security, but they have also driven businesses to innovate their web applications. Over the past few years, cyber criminals have created a new role that is critical for software as a service (SaaS) and other technology businesses. It is now quite common to find directors or managers of product security whose sole responsibility is to make sure any web applications that the development team builds and maintains are safe and secure for customers to use. These individuals protect applications from harmful attacks or unauthorized access. Product security professionals can be found working for the CISO, for the VP of product management, or the VP of development. They participate in engineering projects to identify threats and vulnerabilities, collaborate with the cybersecurity team on the development team's behalf to define security requirements and security concepts, and work with the engineering team to make sure application security is a critical part of your software development lifecycle (SDLC).

Goal Alignment

Product security professionals can help you navigate quite a bit of challenging work across your organization with ease. They are strong advocates for your mission and developing applications quickly. They can also help make sure you build security into the web app development process from the get go. What this means is that you will likely have less rework and vulnerability patching to do, as well as less disruptions to your software development process. Product security professionals are completely vested in securing your business just like your security team, but they understand the web application development process and are there to make sure to align the development team's mission with the security team's mission. This alignment lowers friction between the two teams and creates better outcomes from both perspectives.

You can partner with them early in the SDLC to develop your website and web application with a strong security architecture in mind. They will help you build out security requirements for software products and web applications before they go live and decrease your cyber risk exposure.

Product security professionals will do threat modeling and security analysis of your website and web applications that they will share with you so that you can secure your business together. If you don't have a director or manager of product security at your company yet, this is a new critical role to fill, and we suggest you find one soon. They will be a key ally in your mission to stay ahead of cyberthreats.

07

Conclusion

JavaScript security is becoming a big challenge for many businesses. Organizations struggle to ensure their JavaScript-based websites and web applications are secure and to stay ahead of the latest threats targeting their business and their customers. Unfortunately, traditional security, which focuses primarily on the server side, isn't able to protect businesses or consumers from JavaScript-based attacks. In the meantime, attacks like Magecart, web skimming, and formjacking are wreaking havoc, as businesses struggle to get their arms around client-side security processes, vulnerabilities, and malware.

JavaScript threats have generated an extremely large security gap that businesses must be prepared to address if they wish to continue to operate on the internet. JavaScript web applications make up the client-side attack surface, which is unique and requires its own security program to reduce risk. But it doesn't end there. Web application developers need to be in sync with their security teams to ensure all web applications are maintained and built with security in mind from the start. Client-side protection includes understanding the risks specific to JavaScript web applications and the client side that they run on, cataloging all code associated with web assets and data, properly writing clean code, reviewing the code closely for security issues, applying principles of least privilege and Zero Trust, and ensuring permissions are set correctly.

Comprehensive protection also requires that businesses begin to implement security and web application vulnerability scanning solutions specially designed to protect from JavaScript-based attacks. These solutions are able to analyze JavaScript applications, web browsers, and other client-side technologies in real time. The technologies continuously scan web applications to detect anomalous behavior and then provide crystallized insights and contextual details to web application developers and security analysts to enable them to quickly take joint corrective action. In addition, these products collect client-side telemetry and cyberthreat intelligence that help security teams mitigate cyber attacks on web applications and browsers, further arming your security team to help keep your business and your web apps safe.

It's time for businesses to meet client-side risks and threats with action to enable them to successfully navigate the disruptive forces of cyberattacks.

We are Feroot Security

Feroot Security believes that customers should be able to do business online securely, with any company, without risk or compromise. Our mission is to secure client-side web applications, so our customers can deliver a flawless digital user experience to their customers.

Businesses come to Feroot to enable proactive client-side security programs. Our data protection capabilities take the pain and ambiguity out of client-side security threat analysis, detection, response, and prevention.

Our products help organizations uncover supply chain risks and protect their client-side attack surface.

About Feroot

feroot

Feroot Security believes that customers should be able to do business securely with any company online, without risk or compromise. Feroot secures client-side web applications so businesses can deliver flawless digital user experiences to their customers. Visit www.feroot.com.