LayerX

The Complete Guide to Protecting Against Malicious Browser Extensions

Comprehensive analysis of the risks posed by browser extensions, the key attack vectors, and practical steps to protect organizations against malicious extensions



Table of Contents

Intro: The looming threat of browser extensions **3**

What is a browser extension? **4**
- Browser extension components 5

The Risk of Browser Extensions: 3 Examples of Malicious Exploitation of Permissions **8**
- Example #1: Cookies 8
- Example #2: <all_url> 11
- Example #3: webRequest 13

Unique challenges in mitigating browser extensions' risk **17**
- Extensions are considered harmless software 17
- Misleading users with legitimate names for malicious extensions 17
- Legitimate extensions purchased by attackers 17
- Removal from the web store doesn't disable it in the browser 18
- Users can install malicious extensions unknowingly 18

Browser extensions are a blind spot for traditional security solutions **19**
- Unified Endpoint Management (UEM) 19
- Network security \ SSE 20
- EDR \ XDR \ EPP 20
- Data Leakage Prevention (DLP) 21

Setting the criteria: What is needed for Browser Extension Security **22**

The bottom line: Browser Extension Security **26**

About LayerX browser extension security **27**

INTRO:

The looming threat of browser extensions



Within the ever-changing threat landscape, malicious browser extensions are rapidly claiming presence as a prominent attack vector. While these extensions are perceived by the average user as a legitimate add-on, they also have extensive visibility into - and control over - browser activity and data. Adversaries have not failed to spot this user misconception, as well as the fact that browser extensions are out of the protection scope of solutions like UEM, EDR, SSE, DLP, and all other members of the traditional security stack. As such, the development of malicious extensions that can access and exfiltrate data from browsers and web sessions is nothing but a natural evolution. Given that in most organizations the browser is the main interface to authenticate and access work resources, as well as for creating and interacting with sensitive data, the potential reward of such malicious extensions is immense.



Security professionals are often perplexed when encountering this new browser extensions threat. Indeed, security extensions don't fall into existing threat categories. They are not malware, nor can their data exfiltration be regarded as malicious outbound network traffic. What are they then?



This whitepaper aims to fill this knowledge gap. First, it explains what a browser extension is, what its components are and what control it has over various aspects of the browser. We'll become familiar with the concept of permissions and understand how they play a predominant role for empowering a malicious browser to perform various activities. Following that, we'll delve into the unique challenges that protection against malicious extensions introduces and explain in detail why existing security controls fall short in doing so.



To conclude, we'll build a model of what protection against malicious extensions should look like in order to provide sound mitigation. We'll show how LayerX implements these concepts to deliver a comprehensive protection against this threat.

What is a browser extension?

A browser extension is a piece of software that integrates with standard browsers (Chrome, Safari, etc.), providing them with additional features and functionalities on top of those they natively possess. These functionalities can serve either personal or work uses.

A browser extension is arguably the easiest piece of software to install. All it takes from a user is to browse to their browser vendor's web store and click "install". This high availability makes browser extensions extremely popular among users.

For example, the latest counting shows that there are more than 100K extensions available in the Chrome web store, roughly classified into three types:

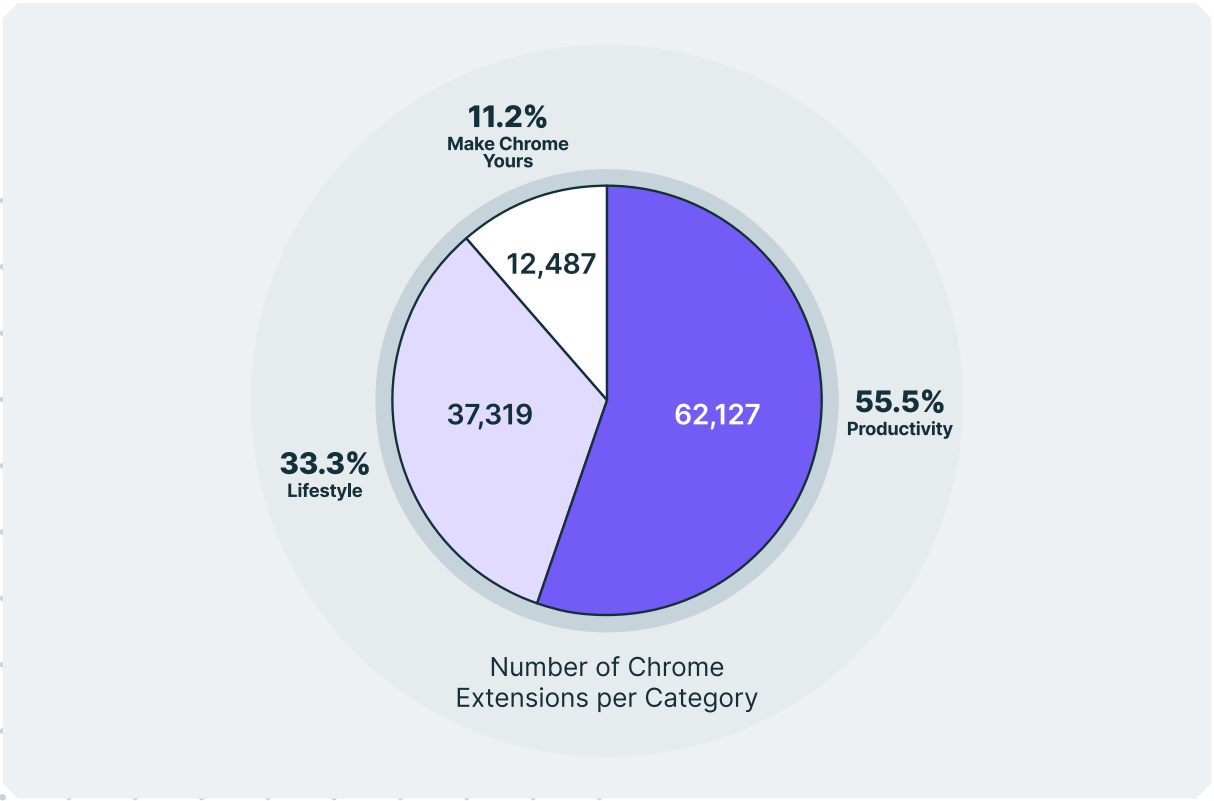


Diagram 1: Number of Chrome extensions per category
Source: https://www.debugbear.com/blog/chrome-extension-statistics

Browser extension components

Let's start our deep dive into the internals of browser extensions. Disclaimer: our aim in this whitepaper is understanding the ways in which a browser extension could act in a malicious manner. In light of that, we'll focus and highlight only what's necessary for that purpose. So this is by no means a definitive analysis of all the complex details an extension developer would need, but rather a guideline for the security practitioner to gain a deeper understanding of the potential browser extension risks.

The browser extension anatomy

Let's start by introducing a simplified diagram of the browser extension's key components and ecosystem. Diagram #2 will serve as a baseline on which we'll later demonstrate how various malicious implementations of browser extensions work.

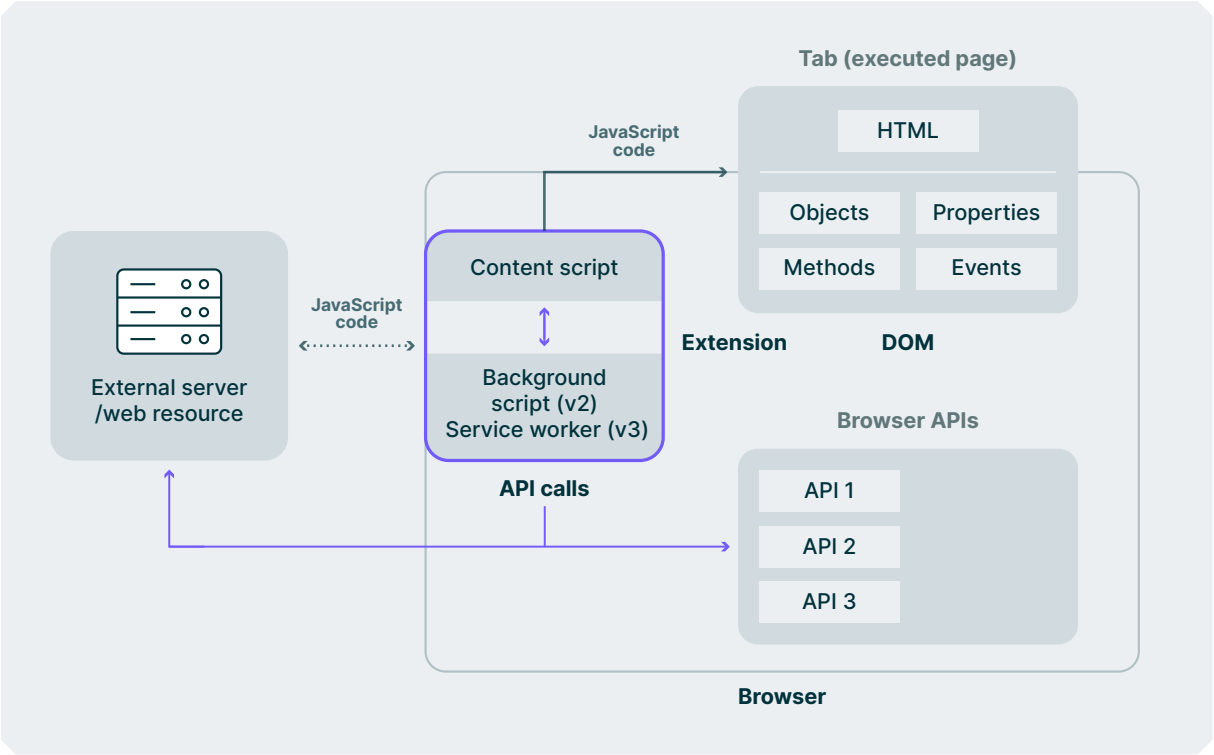


Diagram #2: The browser extension ecosystem

The diagram includes two main objects: the browser itself and the extension within it. There's also an external web resource with which the extension communicates. Within the browser and extension there are also additional objects. Let's understand the relation between these components in more detail.



Background script\service workers

The background script\service workers act as a central hub for handling extension logic. These include managing state, listening for messages, and performing tasks that need broader context, such as network requests or cookie management. Background script\service workers also exchange messages with the content script (see next section).

An important aspect of the background script\service workers is that they are responsible for performing the outbound\inbound network connections required to fetch web pages, install updates, etc. The communication method can take place using the browser API (given that fitting permissions are in place – see next section on permissions) or the JavaScript Fetch and XMLHttpRequest methods.

Service workers were introduced in 2019 as a more granular and efficient replacement for background script. While initially intended to be fully deprecated by January 2024, they are still compatible with Chrome browsers. In this whitepaper we treat both options equally.}

The background script\service workers primarily use the browser's API to perform their operations, though they can also use JavaScript methods for specific purposes.

A browser API is a set of functions and protocols provided by a web browser that allows developers to interact with the browser's features and functionalities, such as handling network requests, accessing browser storage, and more. These APIs enable developers to create dynamic and interactive web applications.}



Content script

The content scripts are typically used to operate on a specific tab by **injecting JavaScript code** into the executed web page **and running it in context**. This code interacts with the page's DOM and can read, modify, or delete elements in the DOM by using standard JavaScript methods. Content script can do that either before the DOM is fully loaded, after the DOM is ready but before the page is loaded, or after the page and all of its entailed resources are loaded. Additionally, the content script can (and often does) exchange messages with the background script\service worker using the browser's API.

Comment

Though the content script is also technically capable of performing outbound communication using JavaScript methods, it is typically the background script\service worker that does this due to the broader context and control it has.

The DOM governs the structure, content, styling, and interactivity of a web page.
It provides a programming interface for developers to manipulate the elements and behavior of a web document dynamically, enhancing user experience and enabling interactive features.}

Browser extension permissions

Browser extension permissions govern and determine the extent to which it is allowed to exercise its power, in terms of functionalities, privacy, and security. **By default, the extension cannot perform any task unless these permissions are explicitly stated.** The permissions (as well the background and content scripts and other parts of the extension that are outside the scope of this discussion) are declared in the manifest file. The manifest file is a JSON file that serves as the extension's configuration files, providing the browser with all the required metadata of the extension.

Aligning with the previous extension anatomy section, we can say that all the different permissions eventually manifest in allowing or restricting the extension's ability to use browser APIs, manipulate the DOM, or access web pages, as illustrated in diagram #3:

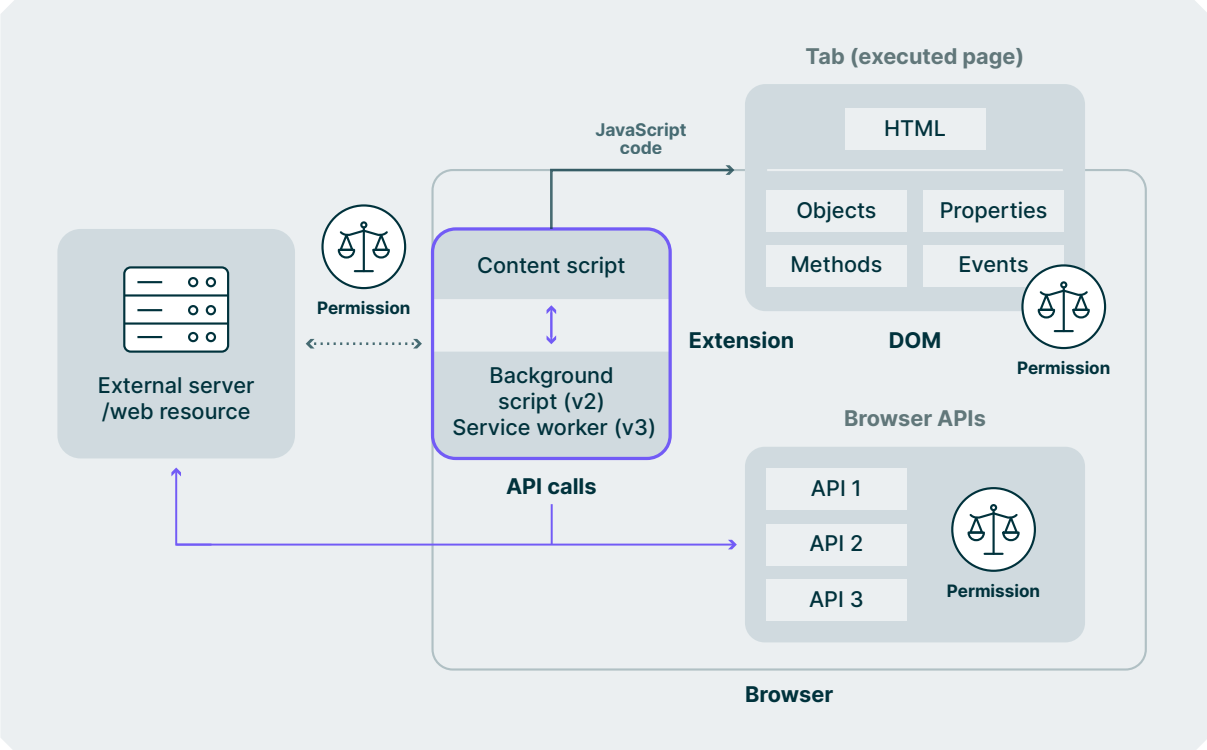
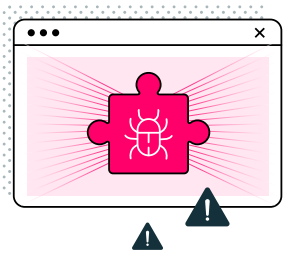


Diagram #3: Extension permissions and their impact

When installing a new extension, the installation prompt outlines to the user the permissions being requested, indicating the level of access the extension will have to the user's web activity.

Naturally, permissions can vary from very restrictive to very allowing. For example, a certain permission might allow the extension to access any web page, while another might restrict it to a predefined group of pages only. As a rule of thumb, the less privileges an extension has, the less of a security issue it is.

Having gained an understanding of what browser extension is, we can proceed to the next chapter. We'll explain how the extensions can also be a grave risk.



The Risk of Browser Extensions: 3 Examples of Malicious Exploitation of Permissions

The risk that browser extensions introduce lies in the permissions extensions request. When abused by attackers, many of these permissions can become the enablers of various malicious operations, from credential theft to user espionage.

In this chapter, we'll illustrate this with several permissions that are in use by common legitimate extensions.

For each, we show what they enable the content or background scripts\service worker to do via DOM manipulation or browser API. We also show how they can be abused for malicious purposes.

Example #1: Cookies

Description

Uses the chrome.cookies API to query and modify cookies and get notified when they change.

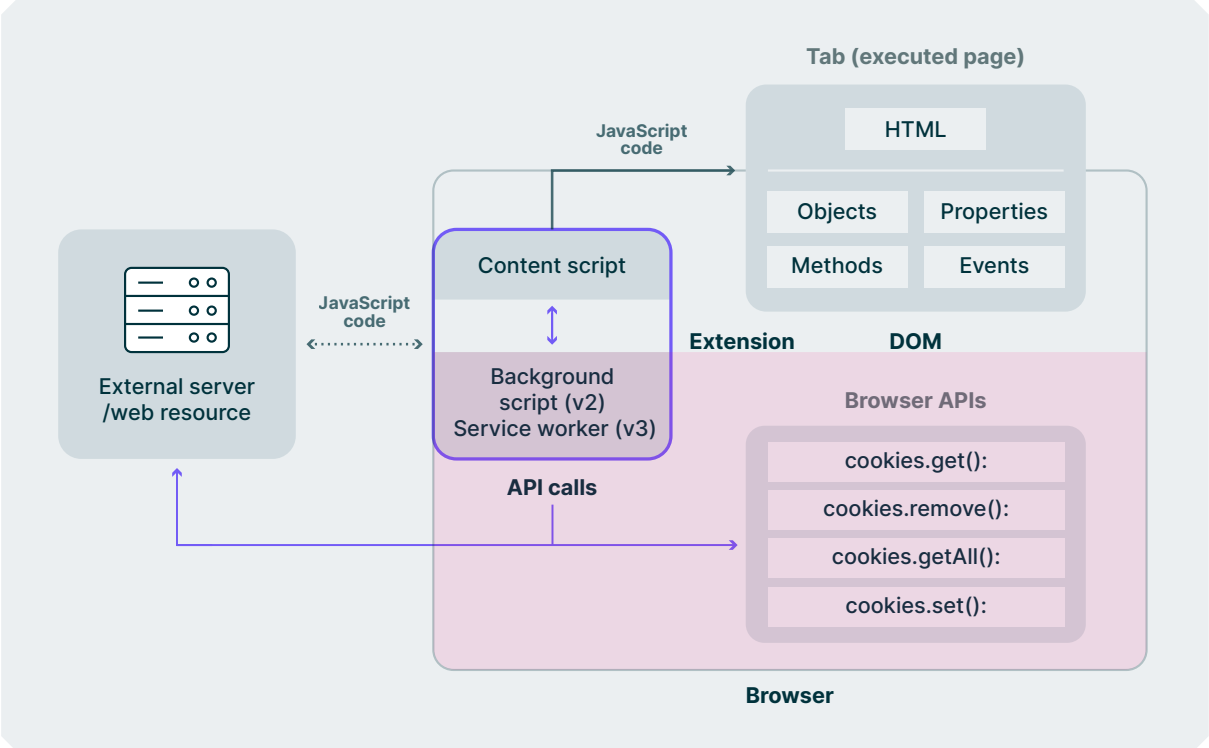


Diagram #4: Cookies permissions and impact

Impact on the extension's components



Content script

No direct access to the browser's cookies API. However, it can communicate with the backend script to trigger the execution of this API, based on predefined events the content scripts identifies in the page.



Background script\service worker

- **Browser API:** The cookies' permissions enable it to access the chrome. cookies API (or its equivalent in other browsers). Specifically:
- **cookies.get():** Retrieve information about a specific cookie by its name, value, or other properties.
- **cookies.getAll():** Access all cookies stored for a particular domain or across multiple domains.
- **cookies.set():** Create or modify a cookie by setting its name, value, domain, expiration, etc.
- **cookies.remove():** Delete a specific cookie by its name and domain.



Direct network access

No impact.

Legitimate purposes for which 'cookies' permissions are required

The cookies' permissions are essential for a variety of legitimate browser extension functionalities, particularly in scenarios where managing, reading, and modifying cookies is integral to the service provided. Common examples would be identity providers' SSO, password managers, online shopping, 2FA, parental control, and others.

Common extensions that require cookies permissions

Okta, Grammarly, Loom.

The cookies query and modification capabilities that the cookies' permissions provide are used by attackers in the following manners:

Account takeover via hijack

The extension retrieves session cookies via the cookies API and exfiltrates them (we'll discuss the required permission for that later on), enabling the attacker

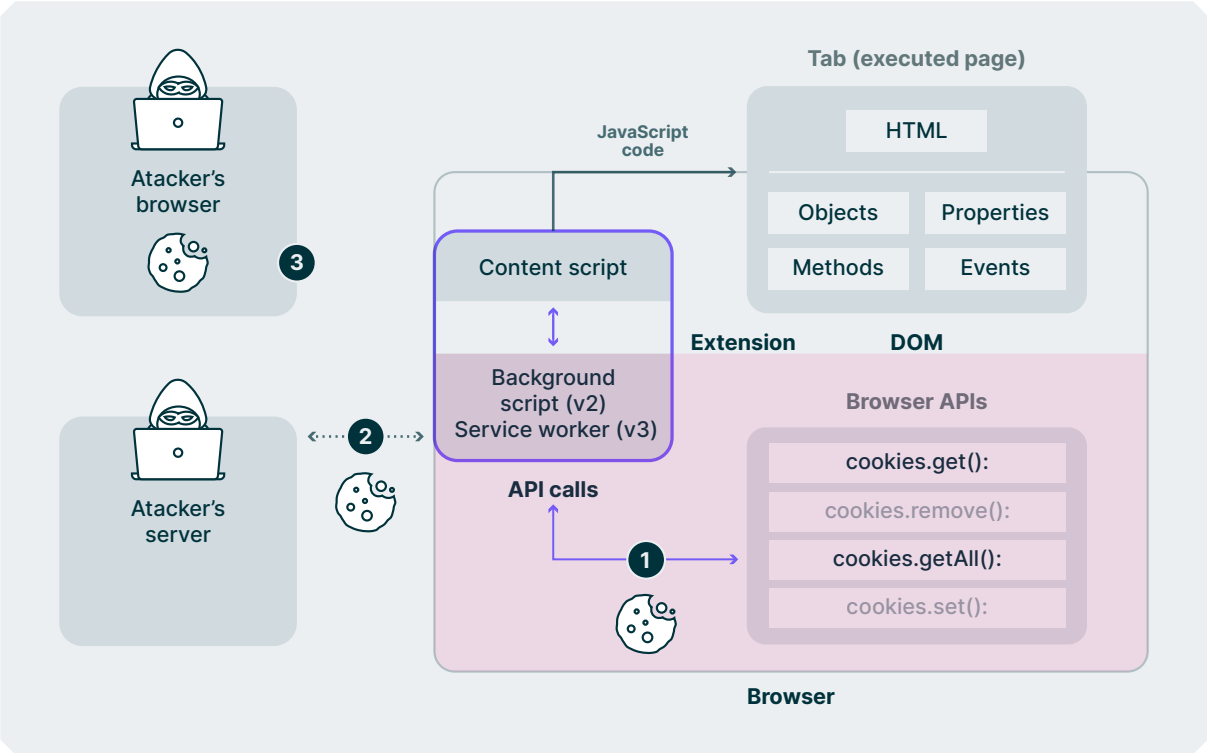


Diagram #5: Session hijacking attack

User cross-site tracking

The extension collects cookies from multiple websites where the user has logged in or visited, creating a detailed web trail profile for the user, severely invading privacy and setting the ground for future attacks.

Cookie injection for exploiting vulnerabilities

An attacker can use a malicious extension to inject specially crafted cookies that exploit vulnerabilities in a website's session management or security features. A prominent example would be to inject a cross-site scripting payload.

Example #2: <all_url>

Description

Grants the extension access to all hosts, enabling reading and changing all data on all websites.

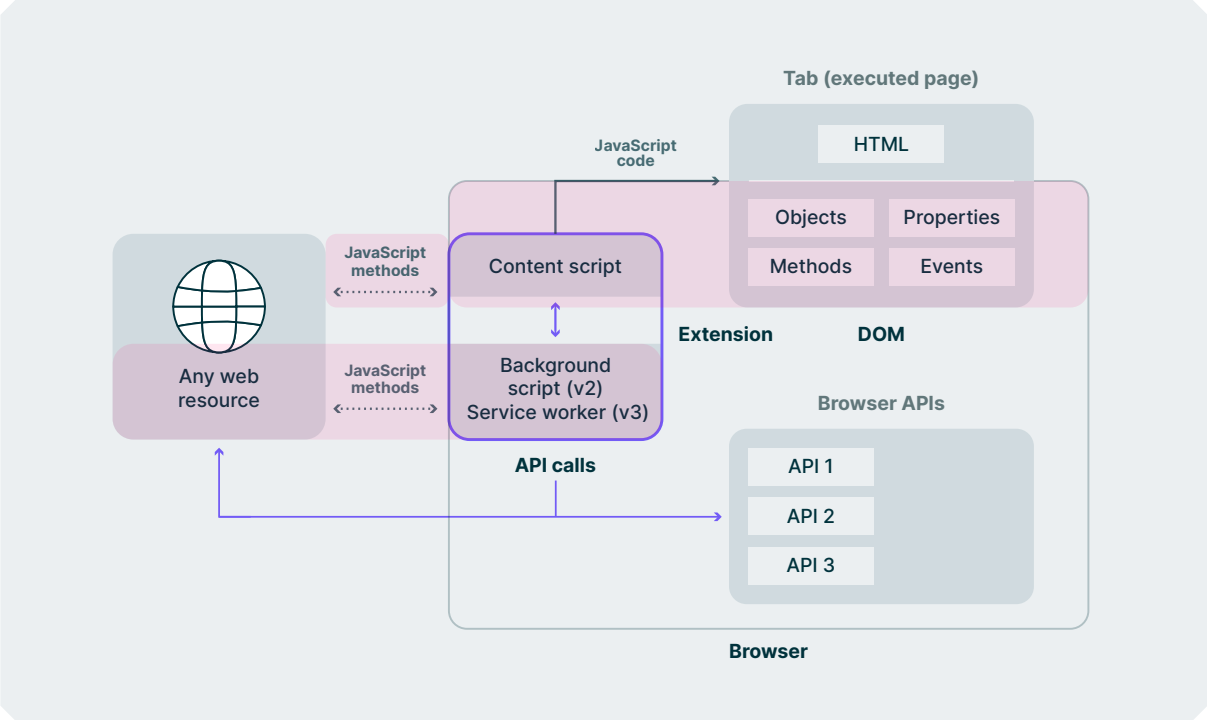


Diagram #6: <all_url> permission impact

Impact on the extension's components

Content script

The primary impact of the <all_urls> permission is that it allows the extension's content script to interact with the DOM of every web page the user visits. This means the script can read, modify, or inject elements into the DOM across all domains, without being limited to specific sites. Without this permission in place the content script can only interact with specific URLs that are predefined in the manifest file.

Background script\service worker

- **Browser API:** The <all_urls> permission doesn't make any additional browser APIs available. However, it expands the scope of other permissions make available to every visited page.
- **Direct network access:** The <all_urls> permission enables the background script\service worker to use JavaScript methods such as XMLHttpRequest and also fetch for outbound connections to any URL.

Legitimate purposes for which '<all_url> is used

The broad access <all_url> provides is necessary mainly for extensions that serve the following purposes: add-blockers, readers, VPN\proxy, form fillers and autofill, security and privacy, and password managers. It's easy to see why – the core value of these types of extensions is their applicability to any web location a user visits.

Common extensions that use the <all_url> permission: LastPass, Google's Autofill, Grammarly, Microsoft Editor.

How attackers abuse the '<all_url> permission

A malicious browser extension, when granted the <all_urls> permission, can exploit its access to web pages with the content script via DOM manipulation to harvest sensitive information, manipulate website functionality, and inject harmful content.

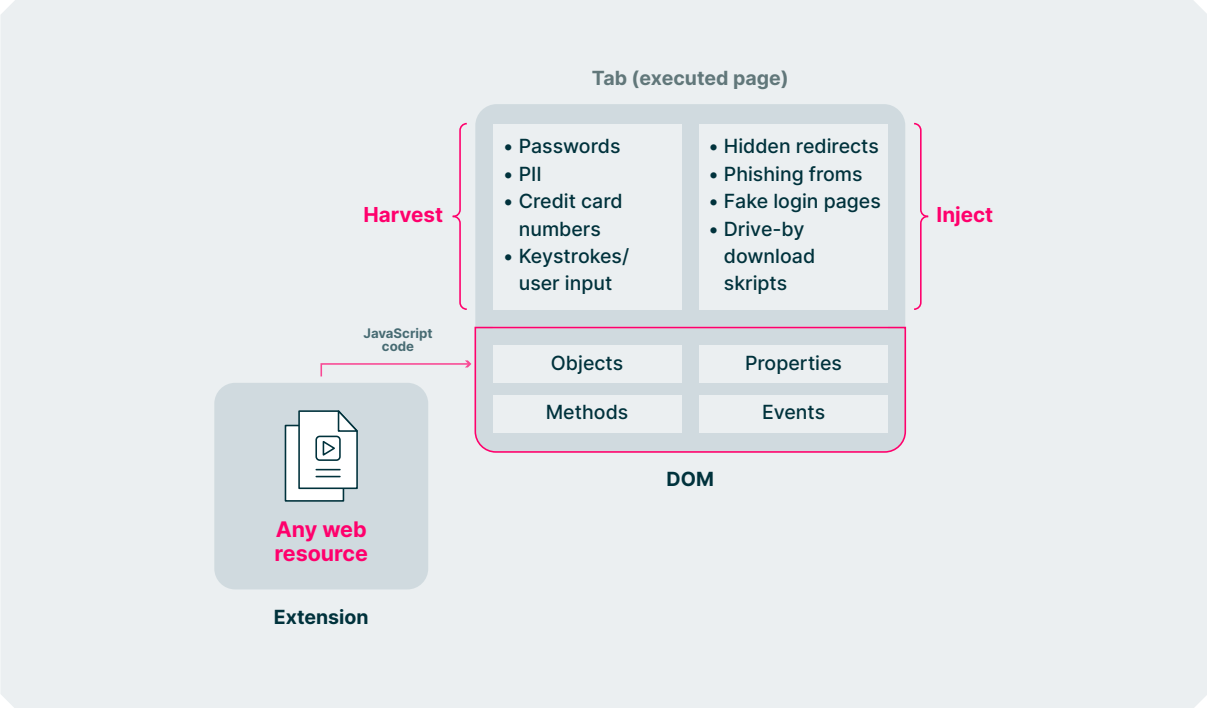


Diagram #7: Malicious abuse of the <all_url> permission

These malicious actions fall into two main groups:

Harvesting sensitive information

Per the attackers' objectives, this data can include everything the web page contains: reading passwords from input fields, logging keystrokes, capturing credit card numbers, and all other types of user input.

Injecting malicious code

This vector can also vary, from hidden redirect buttons that can take visitors to malicious locations such as fake login pages, to phishing forms and drive-by downloads.

Notable examples in the wild

SearchBlox, RoTracker

Example #3: webRequest

Description

Observes and analyzes traffic to intercept, block, or modify requests in-flight. This permission is unique because it can operate on the page before it is fully loaded and rendered in the browser (it can also operate during and after loading).

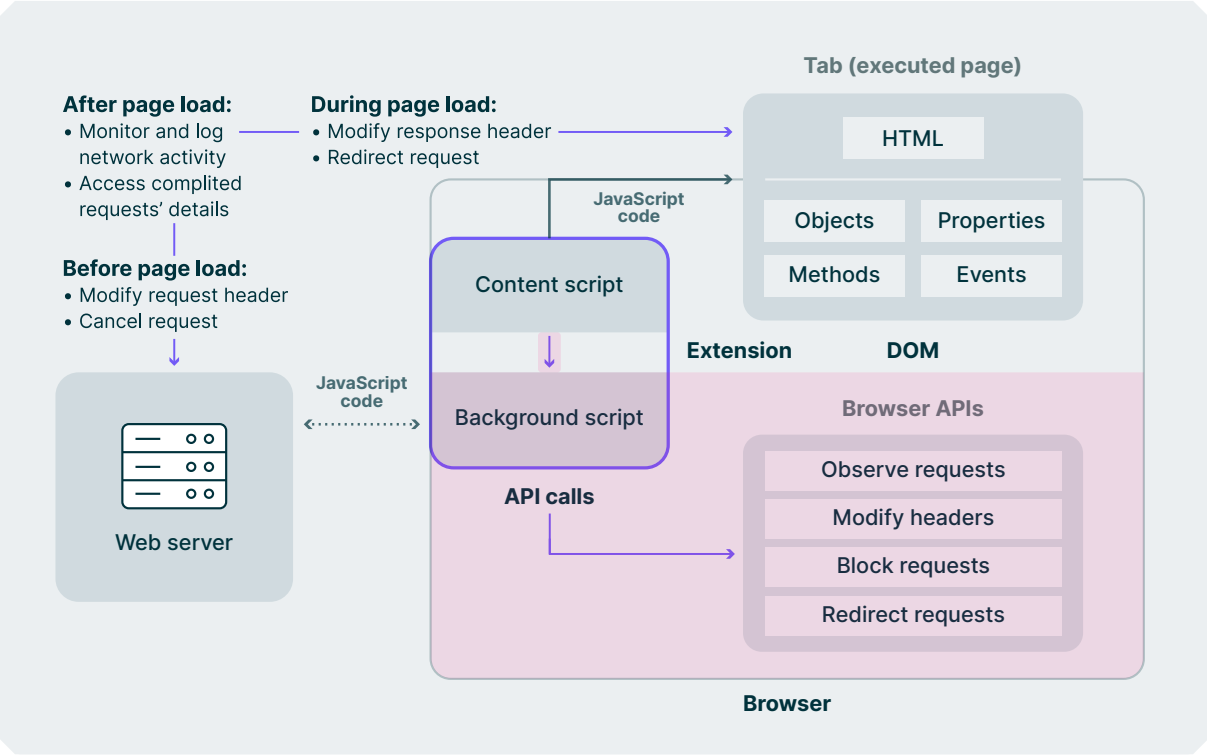


Diagram #8: <webRequest permission impact >

Impact on the extension's components

Content script

While the content script doesn't have access to the webRequest browser API, it can communicate with the background script and instruct it to use said API. For example, as a triggered response to a certain activity within the web page.

Background script\service worker:

Browser API:

- **The Chrome Web Request API:**
  - Observing requests: Using webRequest.onBeforeRequest, a background script\service worker can inspect every network request.
  - Modifying headers: Enables modifying request and response headers with webRequest.onBeforeSendHeaders and webRequest.onHeadersReceived.

- Blocking requests: The webRequest API allows for blocking requests based on certain criteria, such as the URL or HTTP method, using webRequest.onBeforeRequest with a blocking response.
- Redirecting requests: Enables using the API to redirect requests to a different URL

Direct network access:

- **N/A**

Legitimate purposes for which 'webRequest' is used

The webRequest permission in browser extensions is essential for applications that rely on the ability to actively interact with the web traffic. Prominent examples would be ad blockers, privacy enhancers, malware protection, content filters (all types would cancel requests based on domain identification to block access, as well as modify headers if necessary), and proxy\ VPN (mostly with the redirect functionalities).

How attackers abuse the 'webRequest' permission

The webRequest permission can be abused by attackers in various ways depending on the page loading stage:

- **Before the page is loaded**

  Phishing (redirecting users to malicious sites), data exfiltration via URL manipulation, and blocking access to security sites or legitimate content. Special notification should be given to **disabling the Content Security Policy (CSP) feature, which is designed to prevent cross-site scripting (XSS) attacks** by modifying HTTP headers of the page.

- **During page load**

  Man-in-the-middle attacks (injecting malicious scripts), hijacking user sessions, or tracking users by intercepting cookies or tokens, and tampering with response headers (disabling security features).

- **After the page is loaded**

  Exfiltrating data from ongoing network requests, replaying session requests for fraudulent actions, and tracking user behavior by monitoring ongoing network activity.

- **Notable examples in the wild**
  Rlide infostealer, FB stealer

Other notable extension permissions

The table below contains a summary of additional permissions attackers can utilize for various malicious purposes:

Permission	Details	Examples of legitimate extensions	Potential risk
debugger	Instrument network interaction, debug JavaScript, mutate the DOM and CSS, etc., enabling reading and changing all data on all websites.	ull Screenshot, Microsoft Power Automate, Selenium IDE	Malicious extensions can use the debugger permission to access and manipulate the JavaScript code of web pages, potentially injecting malicious scripts.
clipboard	Read data you copy and paste	Office - Enable Copy and Paste, Chrome Remote Desktop, Tampermonkey	Malicious extensions can abuse clipboard access to intercept and steal sensitive data copied by users.
contentSettings	Change settings that control websites' access to features, such as cookies, JavaScript, plugins, geolocation, microphone, camera, etc.	Adblock, FreeVPN	This permission allows extensions to control website content settings, which can be abused to alter security settings or block access to legitimate sites.
desktopCapture\ pageCapture	Capture the content of a screen, individual windows, or tabs.	Loom	These permissions enable extensions to capture the content of the user's desktop or web pages, potentially invading privacy or exposing sensitive information.

Permission	Details	Examples of legitimate extensions	Potential risk
history	Add, remove, and query URLs in the browser's history.	Adobe, Click&Clean	Malicious extensions can access a user's browsing history, compromising their privacy and potentially exposing sensitive information.
privacy	Change your privacy related settings.	ExpressVPN	Extensions with this permission can manipulate privacy settings or track user behavior, leading to privacy violations and data collection without consent.
proxy	Get and set the browser's proxy configuration.	FreeVPN, NordVPN	Malicious extensions can route internet traffic through a proxy server controlled by attackers, potentially exposing sensitive data or redirecting users to malicious websites.
tabCapture	Access a media stream containing video and audio of the current tab.	Screencastify, Loom	This permission can be abused to capture the content of user's browser tabs, potentially invading privacy or exposing sensitive information.
https://*/*	Grant access to all hosts.	Okta, Grammarly, Click&Clean	Extensions with this permission can access all HTTPS websites, making it possible to intercept secure communications and steal sensitive data.



Unique challenges in mitigating browser extensions' risk

Preventing users from installing malicious extensions on their browsers is challenging. In this section we'll shortly review the obstacles to doing so. These are, for the most part, not technical factors, but relate mostly to the relative ease in which a user can be manipulated to install a malicious extension.

1

Extensions are considered harmless software

In the past decade, people have learned to take caution and not download and execute unidentified software. But when it comes to browser extensions, people are less cautious. Extensions are typically perceived as a benign addition to the browser. The fact that the origin of most extensions is from the browser's vendor web store also blurs the distinction between the browser for which the vendor is accountable for and the extension, which can originate from anywhere.

2

Misleading users with legitimate names for malicious extensions

To make things more complicated, attackers can social engineer users with relative ease by associating their extensions' names with known and commonly used software. For example, in 2022 two malicious extensions were spotted, Netflix Party, and Netflix Party 2, which featured 800,000 and 300,000 downloads respectively. These extensions offer functions such as enabling users to watch Netflix shows together, website coupons, and taking screenshots of a website. However, apart from offering the intended functionality, the extensions also track the user's browsing activity by inserting code into eCommerce websites that were visited. This action modified the cookies on the site so that the extension authors received an affiliate payment for any items purchased.

3

Legitimate extensions purchased by attackers

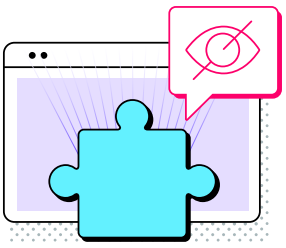
Even more misleading than impersonating the name of a legitimate app or extension, is attackers' ability to purchase a legit extension with an existing install base and modify it for malicious purposes. There's no technical barrier that can withhold an adversary from doing that, nor any way for a user who downloaded the extension from the web store to know that it has been compromised.

4 Removal from the web store doesn't disable it in the browser

Unfortunately, even if the browser vendor identifies an extension as malicious and removes it from its web store, it doesn't have an impact on the extensions that have already been previously installed by users. These extensions continue to operate and execute whatever malicious tasks they were designated to perform even after the store removal.

5 Users can install malicious extensions unknowingly

Aside from direct download from the web store, attackers can take a different route and use various techniques to sideload a malicious extension to users' browsers without their knowledge or consent. Typically, this is done by bundling the extension's download with legitimate software or issuing fake pop ups urging users to update their browsers.



Browser extensions are a blind spot for traditional security solutions

Extensions are the best illustration of a dynamic threat landscape that renders existing solutions obsolete. Indeed, none of the existing security architecture pillars can protect against the malicious browser extensions threat we've outlined in previous sections.

In this section we'll review the four security solutions that are commonly mistaken to be entrusted with protecting against different stages in malicious extensions' lifecycle: download, fetching malicious code from external location, data access, and data exfiltration.

Unified Endpoint Management (UEM): No differentiation between malicious and legit extensions

How it works

UEM, or unified endpoint management, is software that enables IT and security teams to monitor, manage, and secure all of an organization's end-user devices, such as desktops and laptops, smartphones, etc., controlling what software is allowed to be installed.

Assumed protection within the malicious extension kill chain

A UEM could theoretically target the initial download of an extension or continuously scan for installed extensions that violate the UEM's policy.

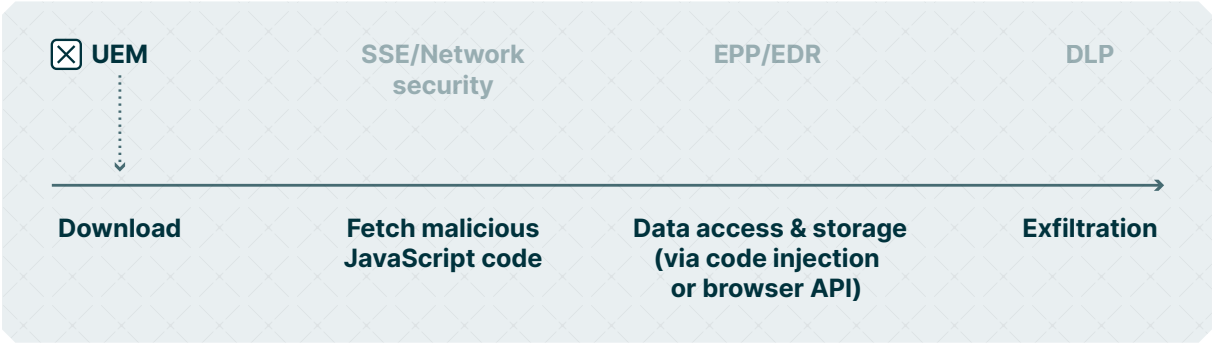


Diagram #9: Browser extension blind spot: UEM

What is the blind spot?

While UEM systems offer a means to deploy and configure browser extensions across enterprise environments, they often fall short in providing sufficient visibility, control, and risk analysis capabilities to discern between malicious extensions and legitimate ones. The only way a UEM can offer protection is by whitelisting a set of extensions and blocking all the rest. This method has failed in all the security fields it has been attempted. Specifically in the case of browser extensions, the disruption in productivity and user experience would be so great, it is unsustainable and ineffective.

Network security \ SSE: No insight into the rendered web page

How it works

SSE solutions inspect inbound and outbound traffic, applying DPI to gain insight into the encrypted network packets. This analysis may reveal external addresses associated with malicious infrastructure or other characteristics of malicious traffic.

Assumed protection within the malicious extension kill chain

Theoretically the SSE would detect the extension's communication with its remote server, either for the purpose of fetching malicious scripts to inject to the page or when exfiltrating the accessed data to a remote location.

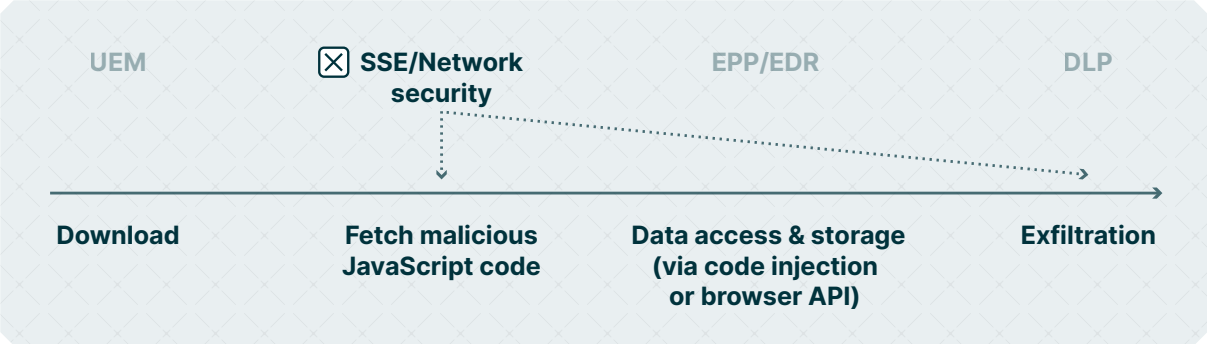


Diagram #10: Browser extension blind spot: SSE\network security

What is the blind spot

SSE solutions don't have any visibility into the application layer in which the web page is loaded and rendered. Additionally, they would typically trust the network traffic initiated by the browser and regard it as legitimate, and would therefore allow it to pass without further inspection.

EDR \ XDR \ EPP: No insight into browser processes

How it works

EPP solutions monitor running processes for anomalous behavior that might indicate malicious intent. These anomalies can be initiating outbound connection, calling OS API, interacting with data files, and other activities that support various parts of an attack's kill chain.

Assumed protection within the malicious extension kill chain

Theoretically the EPP would monitor the executing of the content and background scripts to detect and block malicious activities such as data access and storage.

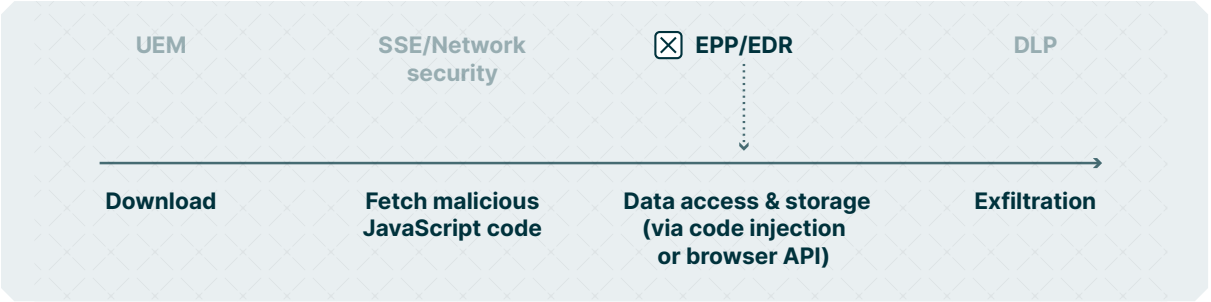


Diagram #11: Browser extension blind spot: Endpoint protection

What is the blind spot?

The EPP implicitly trusts processes that run within the context of the browser and lacks the visibility and analysis to know whether a task within the browser is malicious or benign. So, all tasks performed by the content script or the background script \service worker, including the ones we've outlined in the risks section, are a black box for the EPP.

Data Leakage Prevention (DLP)

How it works

DLP solutions monitor for insecure use of sensitive data files that might lead to exposure. Broadly speaking, they make use of a tagging system that classifies traditional data files (Office, PDF, etc.) based on their sensitivity, and apply respective policies, preventing them from being shared, sent, or downloaded. Certain SaaS-based solutions also prevent the ability of taking screenshots when highly sensitive files are displayed.

Assumed protection within the malicious extension kill chain

Theoretically, DLP would be expected to trigger protection either when sensitive data is accessed and extracted from the page, or when it is exfiltrated outside of the browser's boundaries.

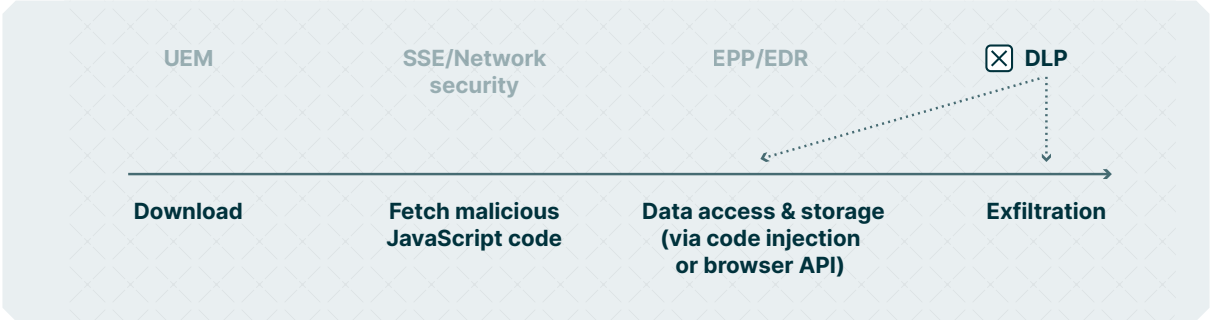


Diagram #12: Browser extension blind spot: DLP

What is the blind spot?

DLP solutions are designed to protect data in file format. As such they offer little to no protection against a malicious extension that captures session data, user activity, or keystrokes. There are no parameters in DLP policies to equip them with the capabilities to identify and mitigate data exposure by this vector.



Setting the criteria: What is needed for Browser Extension Security

In this section we'll outline the capabilities a security solution should possess in order to overcome the obstacles we've outlined in the previous section.

This defense comprises three layers:

Continuous monitoring to discover existing and newly added extensions on the browser.

Risk assessment to determine for each extension if it introduces a potential risk.

Active enforcement that can act and disable extensions that were flagged as malicious.

Let's elaborate more on each.

Continuous monitoring

This is the standard practice every security solution applies to detect malicious behavior within the attack surface it's designated to protect. Relating to the blind spots of existing security solutions we've detailed in the previous section, monitoring is applied to the existence and addition of extensions to the browser on an ongoing basis, to support risk analysis.

Risk assessment

The browser extension's risk assessment should be based on the browser's **attributes** rather than on its activities. There are two reasons for that:

- Browser extensions provide a lot of metadata that can be aggregated for analysis, which can discern between a legitimate extension and a malicious one.
- Focusing on static attributes enables the detection of risks immediately, even when the extension is still idle and hasn't performed any activity yet.

Diagram #13 shows the four-layer funnel of attributes that should be used for a browser extension's risk assessment:

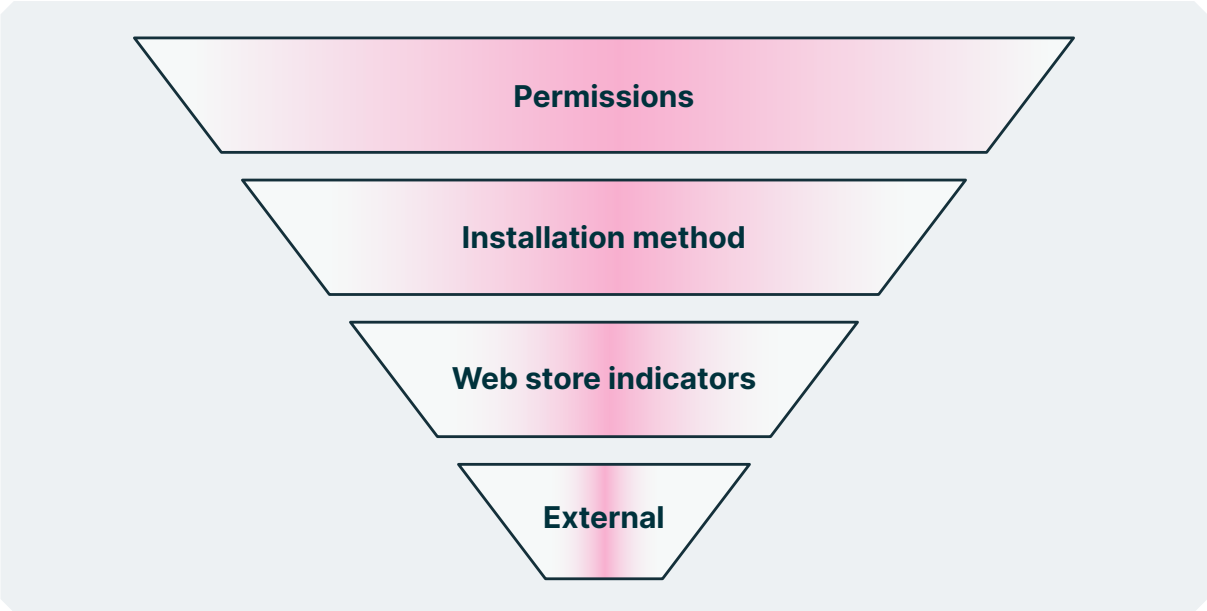


Diagram #13: Browser extension risk analysis funnel

Let's review these attributes in detail.

Permissions

Capability

Visibility into the extension's permissions and rating its potential malicious use based on the aggregation of those permissions.

As we've explained earlier in detail, the extension's permissions determine its scope of capabilities. Any permission, or a combination of permissions, that enables an extension to perform any of the malicious actions we've listed in the previous sections should render it marked as a potential risk.

However, we've also seen that multiple legitimate extensions make use of these permissions as well. Therefore, we need additional criteria on top of the mere permission list.

Installation method

Capability

Visibility into the extension's installation method and rating its implication on the extension's origin. A browser extension can be installed on a browser in various manners. From the most to least secure:

- **Admin distribution**
  Extensions that are broadly installed across employees' browsers as part of the organization's software. These extensions are typically vetted and approved by the IT and security teams.
- **Personal download**
  Extensions that were downloaded from the web store by individual users for either work or personal purposes. These extensions might or might not adhere to the organization's security policies.
- **Developer upload**
  Extensions that weren't downloaded from a web store but loaded by employees locally. These extensions are even further under the radar of the organization's security policies.

- **Sideload**
  Extensions that were installed by a third party app and not by the employee directly. This method is the least secure as adversaries could easily abuse it to install malicious extensions on users' browsers without their knowledge and consent.

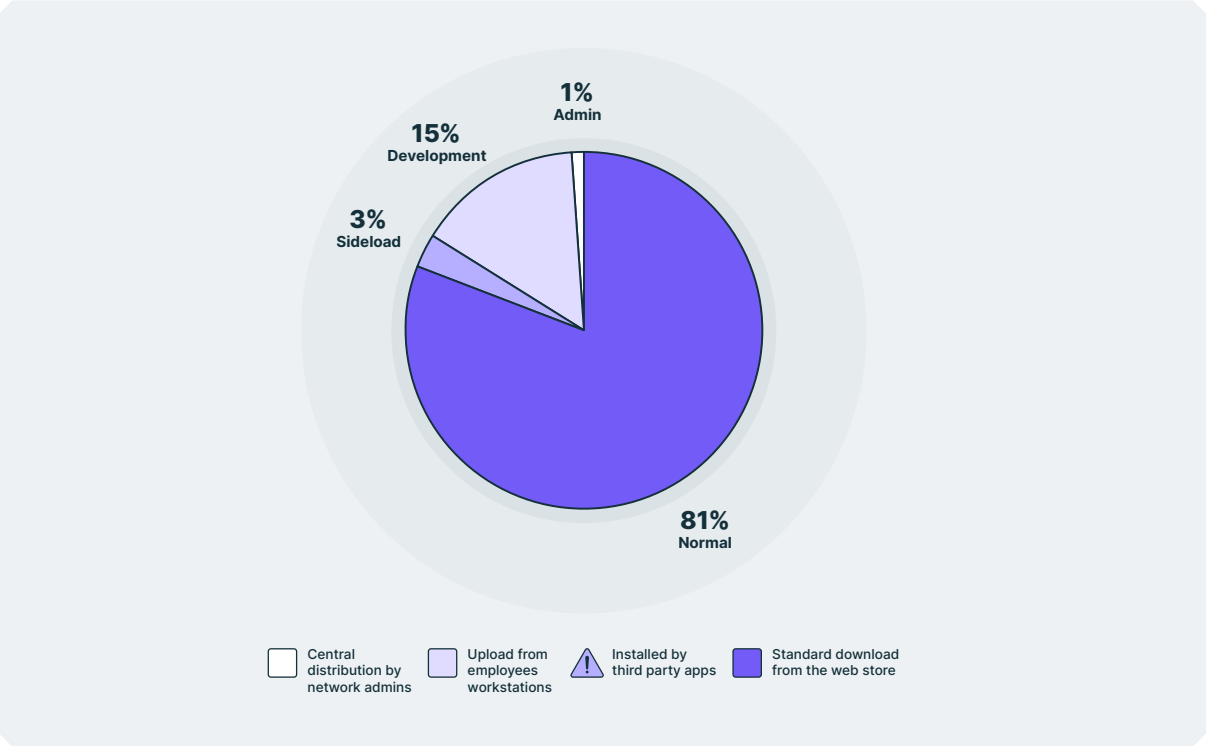


Diagram #13: Browser extension risk analysis funnel

The installation method serves as a reliable filter to determine whether an extension with high permission is risky or requires an additional analysis.

Web store indication

The information that is available in the web store about an extension can provide insights into its security posture. As a rule of thumb, the more an extension is widely used in a standardized manner, the safer it is to download. Counter to that, the following list shows which extension attributes should raise caution:

- **Address and Email**
  If the developer's contact address or email is missing from the Chrome Web Store listing, this could indicate potential obscurity and a lack of accountability.
- **Last Updated**
  How long ago the extension was last updated, reflecting the potential security and compatibility risks associated with outdated extensions.
- **Privacy Policy**
  If an extension lacks a privacy policy in its Web Store listing, this could indicate potential concerns about data handling and user privacy.
- **Rating**
  The extension's user ratings, assessing the extension's overall quality and user satisfaction.

- **Number of Rating Users**
  The overall number of user ratings at the time of assessment. The more ratings, the smaller the risk.
- **Support Site**
  No support site associated with the extension on the Web Store could suggest potential challenges in receiving assistance or troubleshooting issues.
- **Users**
  The overall number of users. A low number of users may impact support and indicate lower reliability.
- **Website**
  No website associated with the extension, potentially signifies a lack of additional information or resources.

Other external indications

These are additional indicators that, when present, can also add to the overall risk scoring of the extension. These can include for example:

- The extension doesn't appear in any official store.
- The extension is promoted either for free, or in any other way that doesn't make economic sense. For example free extensions pushed by paid ads

The decision tree in diagram #15 shows a simplified form of the risk analysis process that would determine whether a given extension is malicious or legitimate.

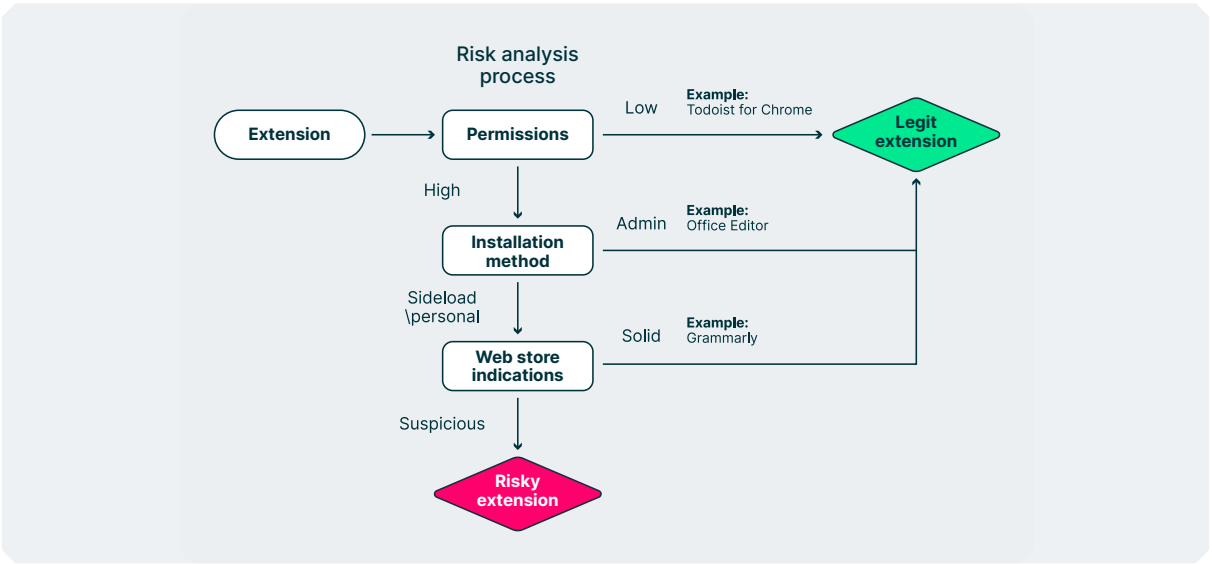


Diagram #15: Browser extension risk analysis decision flow

Browser extension active enforcement

The active enforcement component is required for the risk mitigation of extensions that were analyzed and flagged as malicious. Theoretically, this could be implemented either by disabling the extension or removing it altogether. However, for operational and pragmatic reasons, disabling is the preferred option, so the actual removal would be done manually and knowingly by the users or their admins rather than automatically.

The bottom line: Browser Extension Security

Browser extensions are a ubiquitous part of the modern browsing experience. Expecting users to give them up is unreasonable, and they can be helpful tools that enhance productivity and user experience.

However, organizations need to recgonize that they also pose a significant cybersecurity risk. These small pieces of software can access sensitive browser activities and data, making them a potential vector for malicious exploitation. Attackers can abuse the permissions granted to extensions, such as accessing cookies, tracking user activity across sites, harvesting information, or even injecting harmful code. At the same time, users often aren't aware of their malicious potential and web stores cannot uninstall a malicious extension from the users' browsers, making them easy for attackers to distribute, operate, and exploit.

Traditional security solutions like UEMs, network security tools, or EPPs struggle to detect and mitigate these threats due to the unique and overlooked nature of extensions. Addressing this gap requires a dedicated browser extension security solution capable of continuous monitoring, risk assessment, and active enforcement.

LayerX provides such a solution by offering full protection across the entire lifecycle of browser extensions. LayerX ensures visibility into existing and newly installed extensions, evaluates their permissions and origins, can disable extensions, and prevents potentially malicious actions before they occur.

Those advanced capabilities enable organizations to proactively safeguard their environments from this evolving threat vector while maintaining user productivity and security.

About LayerX
One Browser Extension to Rule Them All

Comprehensive Audit

Discover all extensions on all browsers for all users, with full visibility and control

Rich Risk Classification

Assess the risk profile of each extension using internal and external risk factors

Adaptive Enforcement

Go beyond manual blocklists to automatically disable or block extensions based on their risk

0% User Friction

Easy deployment with no impact on the user browsing experience or existing workflows

LayerX browser security platform provides full protection against malicious browser extensions. LayerX's secure browser extension can integrate with any browser, and as such has full visibility into all other installed extensions.

LayerX's extension continuously monitors the existing and newly installed extensions, evaluating permissions, installation method, web store parameters, and external risk parameters.

LayerX identifies risky browser extensions using a comprehensive risk-scoring approach that combines internal and external risk factors. LayerX examines the access permissions requested by each extension and whether it has access to sensitive information such as passwords, cookies, user input, and more. At the same time, LayerX analyzes the extension's reputation based on external factors such as user rating, number of downloads, age, and more. These parameters are combined to create a unified score reflecting each extension's risk.



With its granular policy engine, LayerX enables its users to trigger notifications, alerts, or even complete disablement of an extension, when any risk indicator or combination of these are detected. LayerX extension runs at a higher permission level than ordinary extensions, and cannot be tampered with or uninstalled by users.

To learn more about how LayerX can help you manage and secure your browser extensions, go to www.layerxsecurity.com and schedule a demo today!