آسیب‌پذیری‌ها آلودگی کش وب[1]

تاریخ گزارش: ۲۳ تیر ۱۳۹۸

تهیه شده توسط تیم فنی آزمایشگاه امنیت کی‌پاد

خلاصه

آسیب‌پذیری آلودگی کش وب (Web Cache) برای مدت طولانی به عنوان یک تهدید نظری[2] تصور شده بود. با این حال، اکنون پژوهش‌هایی در حوزه امنیت سامانه‌های کامپیوتری انتشار پیدا کرده است که در آن این آسیب‌پذیری به صورت عملی مورد بهره‌برداری قرار گرفته است.

علاوه بر اینکه، حملات بر علیه آسیب‌پذیری‌های آلودگی کش وب به صورت عملی توسط پژوهشگران پیاده‌سازی شده است، اکنون نمونه‌هایی از این حمله هم در محیط واقعی و عملیاتی مشاهده شده است و توسط آزمایشگاه امنیت کی‌پاد مورد رصد قرار گرفته است.

از همین روی، در این مقاله به آسیب‌پذیری آلودگی کش وب خواهیم پرداخت تا قبل از اینکه توسط مهاجمین مورد بهره‌برداری گسترده قرار بگیرد، رویکردهای امنیتی برای مقابله با این حملات تعیین و همچنین به صورت عملی اتخاذ شود.

در این مقاله ابتدا آسیب‌پذیری کش وب را بررسی خواهیم کرد که چگونه می‌توان با استفاده از ویژگی‌های محرمانه محیط وب حافظه کش را به یک سامانه بهره‌برداری (Exploitation) تبدیل کنیم و هر کسی که یک صفحه آلوده را بازدید می‌کند، مورد نفوذ قرار بدهیم.

کلیدواژه:

آسیب‌پذیری آلودگی کش وب، حملات جدید وب، کش وب، آسیب‌پذیری وب

[1] Web Cache Poisoning Vulnerabilities

[2] Theoretical Threat

جزئیات فنی

مکانیزم کش[3] اطلاعات درون حافظه با کاهش تاخیر[4] و همچنین کاهش بار پردازش در سمت سرور موجب بهبود مدت زمان بارگزاری صفحات وب می‌شود. شایان ذکر است، مکانیزم کش وب می‌تواند در سطوح مختلف از قبیل محصولات نرم‌افزاری خاص، ساختارهای CDN، درون خود برنامه‌های تحت وب و فریمورک‌ها پیاده سازی شود. به همین دلیل، تمامی این سطوح مستعد به حمله آلودگی کش وب هستند.

قابل ذکر است، آلودگی کش وب یک نوع خاص از خانواده آسیب‌پذیری آلودگی کش است. به هر صورت، در این مقاله نشان خواهیم داد که چطور با استفاده از این آسیب‌پذیری به عنوان یک مهاجم می‌توانیم کنترل بر روی وب‌سایت‌ها و فریمورک‌ها به دست آوریم و از انجام حملات تک درخواستی (Single-Request Attacks) به پیاده‌سازی زنجیره‌ای از بهره‌برداری برسیم. در نهایت هم به این موضوع خواهیم پرداخت که چگونه می‌توانیم در مقابل این حملات از خود دفاع کنیم.

تاثیرپذیری:

کش سرورها و سرویس‌ها، برنامه‌های تحت وب، فریمورک‌ها

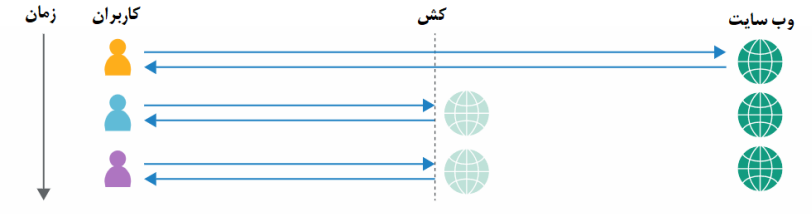
مفاهیم اصلی

برای درک آسیب‌پذیری آلودگی کش، ابتدا باید یک نگاهی به مکانیزم کش منابع در فضای وب بندازیم که چگونه این مکانیزم کار می‌کند و در نهایت چگونه این مکانیزم باعث بهبود بارگزاری صفحات وب می‌شود.

[3] Caching

[4] Latency

مکانیزم کش در بین کاربر و برنامه سرور رخ می‌دهد، جایی که کپی برخی از درخواست‌ها (Requests) ذخیره و مجدد به کاربران جدید ارائه می‌شوند. در تصویر ۱، سه کاربر را مشاهده خواهید کرد که در حال واکشی یک منبع مشابه از سرور هستند.



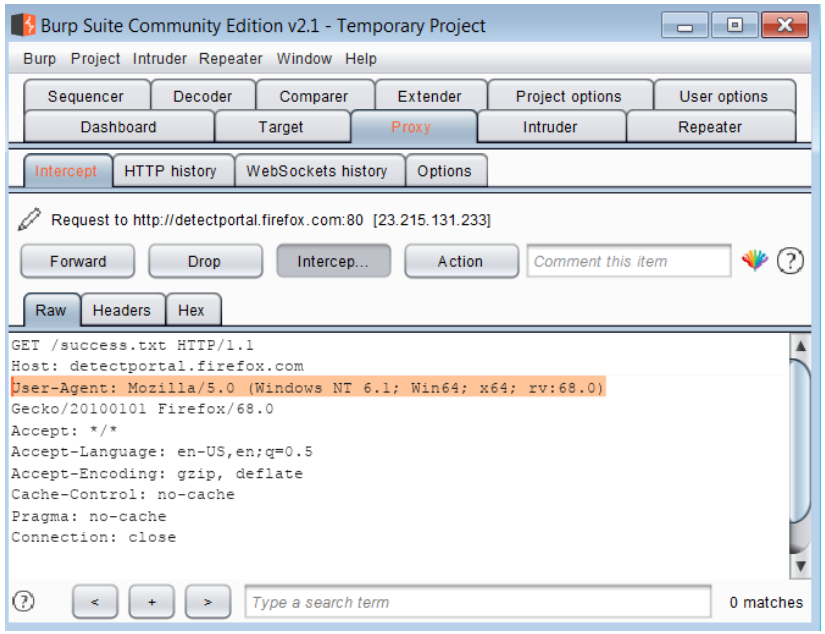
تصویر ۱: دیاگرام مکانیزم کش اطلاعات/منابع

مکانیزم کش در نظر گرفته شده است تا سرعت بارگزاری صفحات وب را با کاهش زمان تاخیر افزایش دهد و همچنین بار پردازش در سمت سرور را هم کم کند. برخی از شرکت‌ها کش خود را با استفاده از نرم‌افزارهایی مانند Varnish میزبانی می‌کنند و برخی دیگر برای کش پراکنده در سطوح مختلف جغرافیایی متکی به ساختارهای CDN مانند Cloudflare هستند. اگر چه برخی از برنامه‌های تحت وب و فریمورک‌های معروف مانند Drupal یک مکانیزم کش داخلی[۵] دارند. شایان ذکر است، انواع دیگر از رویکردها برای کش اطلاعات/منابع مانند کش‌های سمت کاربر[۶]، کش‌های مبتنی بر DNS[۷] و ... هم وجود دارد که در این مقاله به آن‌ها نخواهیم پرداخت.

کلیدهای کش[۸]

مسئله این است که مفهوم کش شاید ساده و سرراست به نظر برسد، اما چندین فرضیه ریسکی در آن وجود دارد. به عنوان مثال، هرگاه کش یک درخواست برای ارائه منبعی را دریافت می‌کند، در ادامه نیاز دارد تصمیم بگیرد که آیا از آن منبع یک کپی ذخیره شده دارد یا باید درخواست برای منبع را به خود سرور عبور دهد.

شناسایی این مورد که آیا دو درخواست تلاش به بارگزاری یک منبع مشابه را دارند، می‌تواند عملا یک مسئله دشوار باشد زیرا نیاز است درخواست ها بایت به بایت با هم دیگر شباهت داشته باشند که این مورد عملا غیرموثر است، چون درخواست‌های HTTP دارای داده‌های ناخواسته و بسیار متغییری هستند از قبیل نوع مرورگری که درخواست را ایجاد کرده است:



تصویر ۲: ساختار یک درخواست HTTP

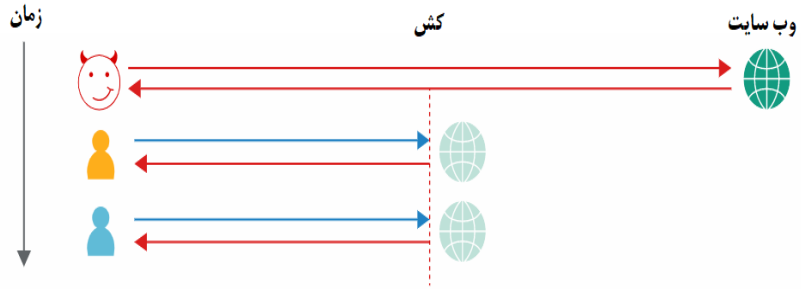
در مکانیزم کش داده‌ها و منابع، این مشکل با استفاده از مفهومی به عنوان کلیدهای کش رفع می‌شود. در این مکانیزم فقط چندین کامپوننت مخصوص از یک درخواست HTTP برای شناسایی منبع درخواست شده مورد استفاده قرار می‌گیرد. به عنوان مثال، در درخواست بالا مقدار GET و Host برای تشخیص یک منبع مورد استفاده قرار گرفته است. از همین روی، اگر کاربر دیگری درخواست با مقادیر مشابه در این پارامتر (تصویر ۲) ایجاد کند، کش تشخیص

[۵] Built-in
[۶] Client-Side Cache
[۷] DNS Caches
[۸] Caches Keys

می‌دهد که باید داده از پیش کش شده را ارائه کند، چون منبع درخواست شده توسط دو کاربر مشابه یک دیگر است. ولی اگر درخواستی مشابه با درخواست بالا (تصویر ۱) ایجاد شود که به عنوان مثال پارامتر زبان آن به جای انگلیسی، زبان فارسی باشد به طور کل چیزی که بارگزاری خواهد شد، با چیزی که درخواست داده شده است، متفاوت و اشتباه خواهد بود.

آلودگی کش

هدف از آلودگی کش وب ارسال یک درخواست است که به موجب آن یک واکنش[۹] مخرب ایجاد خواهد شد که در کش ذخیره شده است و به کاربران دیگر ارائه می‌شود.



تصویر ۳: آلودگی کش

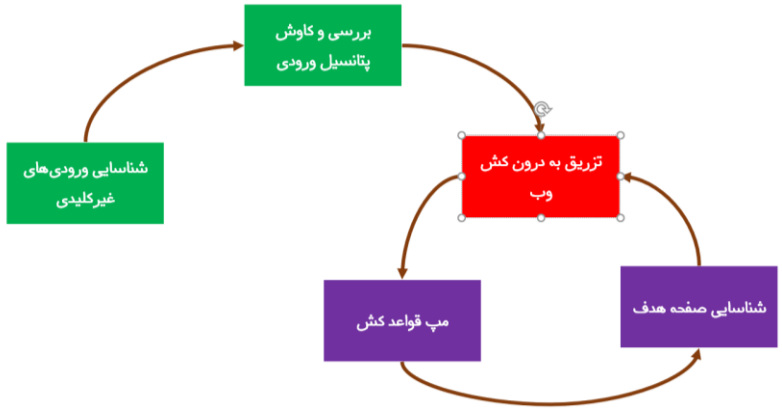
در این مقاله، قصد ما بر این است که کش را با استفاده از ورودی‌های غیرکلیدی[۱۰] مانند هدرهای HTTP آلوده کنیم. البته شایان ذکر است، این روش تنها روش ممکن برای آلوده‌سازی کش نیست، روش‌های دیگری مانند HTTP Response Splitting و Request Smuggling هم وجود دارند که می‌توان از آن‌ها برای آلوده‌سازی کش بهره برد.

همچنین به این نکته توجه داشته باشید که کش وب را می‌توان با روش کاملا متفاوت دیگری مورد حمله قرار داد که این حمله با عنوان Web Cache Deception شناخته می‌شود که با حملات آلوده‌سازی کش وب نباید اشتباه گرفته شود.

[۹] Response

متدلوژی

به منظور شناسایی و کشف آسیب‌پذیری آلودگی کش وب از متدلوژی زیر می‌توان استفاده کرد که در قالب تصویر ۴ به نمایش در آمده است.



تصویر ۴: کشف و شناسایی آلودگی کش

در ادامه این مراحل با جزئیات توضیح داده خواهد شد، ولی با این حال در این قسمت یک بازبینی کلی از کل مراحل ارائه می‌شود. همانطور که در دیاگرام نمایش داده شد، اولین مرحله شناسایی ورودی‌های غیرکلیدی است.

انجام این مرحله به صورت دستی کمی دشوار و سخت است، اما برای خودکارسازی این مرحله می‌توان از یک افزونه برای ابزار Burp با عنوان Param Miner استفاده کرد که این مرحله را به صورت خودکار با حدس نام‌های کوکی/هدر انجام می‌دهد. این افزونه با بررسی کوکی‌ها و هدرهای متفاوت و در ادامه بررسی واکنش آن‌ها بر روی صفحه وب ورودی‌های غیرکلیدی را تشخیص می‌دهد.

بعد از شناسایی ورودی‌های غیرکلیدی، گام بعدی ارزیابی این مسئله است که چه میزان خسارت می‌توان با استفاده از آن‌ها به برنامه وب وارد کرد، سپس در گام بعد تلاش به ذخیره آن‌ها در کش خواهد شد. اگر این ذخیره‌سازی شکست بخورد، شما نیاز خواهید داشت، دانش عمیق‌تری

[۱۰] Unkeyd Input

از چگونگی کارکرد کش به دست آورید و در ادامه یک صفحه قابل کش را شناسایی کنید.
شایان ذکر است، صفحه وب برای اینکه کش شود به فاکتورهای بسیار زیادی از قبیل فرمت فایل[۱۱]، نوع محتوا[۱۲]، مسیر[۱۳]، کد وضعیت[۱۴]، و هدرهای واکنش[۱۵] نیازمند است.
واکنش‌های کش شده می‌توانند ورودی‌های غیرکلیدی را پنهان کنند، بنابراین اگر تلاش به شناسایی و کاوش دستی ورودی‌های غیرکلیدی کنیم، اصطلاحا به یک cache-buster نیاز است. با این حال اگر افزونه Param Miner در Burp به درستی بارگزاری شده باشد، می‌توانید با افزودن یک پارامتر به همراه مقدار $randomplz به کوئری استرینگ « String Query» اطمینان حاصل کنید که هر درخواست دارای یک کلید کش منحصربفرد است.
هنگام ارزیابی یک وب‌سایت، آلوده کردن کاربران دیگر یک خطر همیشگی است. از همین روی، افزونه Param Miner با افزودن یک cache-buster به تمامی درخواست‌های Burp خطر این مسئله را کاهش می‌دهد. این cache-buster دارای یک مقدار ثابت است، بنابرین می‌توان رفتار کش را برای خود مشاهده کرد، بدون اینکه کاربران دیگر تحت تاثیر قرار بگیرند.

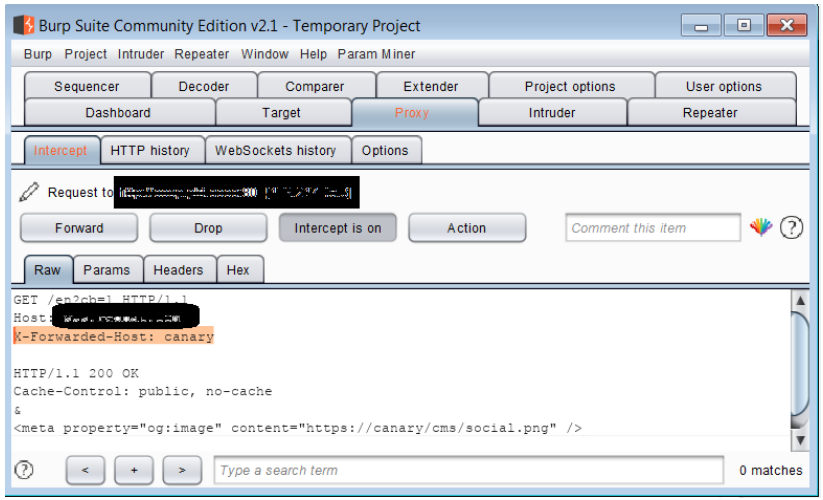
موارد مطالعاتی

حال اجازه بدهید این مسئله را بررسی کنیم که اگر این موارد نظری را بر علیه یک وب‌سایت واقعی و عملیاتی مورد استفاده قرار بدهیم، چه اتفاقی رخ خواهد داد. از آنجایی که وب‌سایت بررسی شده هنوز در زمان انتشار مقاله آسیب‌پذیر هست، آدرس این وب‌سایت ذکر نخواهد شد تا برای کاربران آن‌ها تهدید جدی رخ ندهد. ولی در هر صورت، به صورت کلی نمایش داده خواهد شد که این آسیب‌پذیری به چه شکل کار می‌کند و نتیجه نهایی حمله و بهره‌برداری از این آسیب‌پذیری چیست.

[۱۱] File extension
[۱۲] content-type
[۱۳] Route

آلوده‌سازی کش مقدماتی

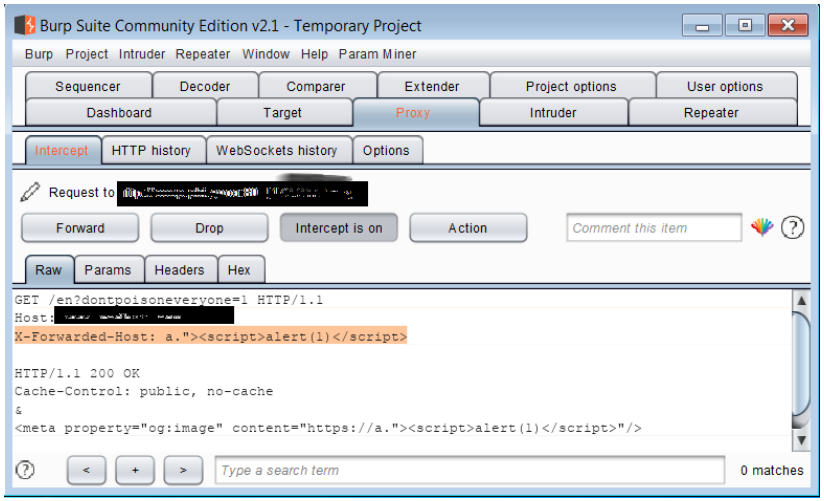
آلودگی کش اغلب اوقات به سادگی قابل بهره‌برداری است. برای شروع این پروسه، به وبسایت هدف رجوع خواهیم کرد که بلافاصله افزونه Param Miner یک ورودی غیرکلیدی را شناسایی می‌کند.



تصویر ۵: شناسایی ورودی غیرکلیدی توسط Param Miner

در اینجا می‌توانیم مشاهده کنیم که هدر X-Forwarded-Host توسط برنامه برای ایجاد Open Graph URL درون تگ meta استفاده شده است. در گام بعد باید بررسی شود که آیا این مسئله قابل بهره‌برداری است یا خیر. به همین دلیل، در گام بعد یک پیلود XSS ساده مورد استفاده قرار خواهد گرفت.

[۱۴] Status Code
[۱۵] Responsive Headers



تصویر ۶: استفاده از یک پیلود XSS

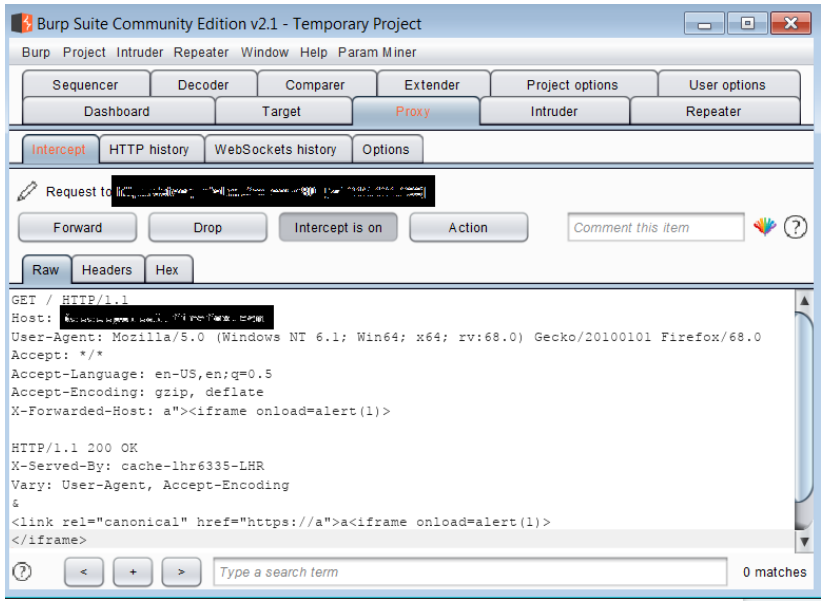
خوب تا به اینجا مشخص شد که می‌توانیم پاسخی ایجاد کنیم که هر کس آن را دریافت کرد بر روی مرورگر آن یک کد جاواسکریپت اجرا شود. در گام بعد باید بررسی شود که اگر این پاسخ درون کش ذخیره شود، قابل انتقال به دیگر کاربران است یا خیر.

البته نباید تصور کرد که صرفا با تعریف شدن Cache Control: no-cache در پیکربندی سرور، دیگر این حمله ممکن نیست. همیشه بهترین گزینه برای آزمایش امنیت این است که فرض شود حمله کار نخواهد کرد. از همین روی، ابتدا می‌توانیم با ارسال مجدد درخواست بدون هدرهای مخرب مشکل را تائید کنیم و سپس از یک ماشین و مرورگر دیگر و فراخوانی مستقیم URL اقدام به بهره‌برداری از آسیب‌پذیری کنیم.

با بررسی که صورت گرفت مشخص شد که این وب‌سایت آسیب‌پذیر به آلودگی کش است که آدرس آن و دیگر جزئیات هدرهای آن در Burp حذف و پنهان شده است.

آلوده‌سازی انتخابی[۱۶]

شایان ذکر است، هدرهای HTTP می‌توانند اطلاعات مهم دیگری را ارائه دهند که مشخص کند نحوه عملکرد داخلی کش به چه شکل است. به عنوان مثال، نمونه نمایش داده شده در تصویر ۷ را در نظر بگیرید.



تصویر ۷: آلودگی کش انتخابی

اطلاعات نمایش داده شده درون هدرها ابتدا مشابه با مثال اول به نظر خواهند رسید. با این حال، هدر به ما می‌گوید که User-Agent ممکن است قسمتی از کلید کش باشد و البته بررسی دستی این مورد تائید کننده این مدعا بود.

این بدین معنا است که ما مدعی استفاده از فایرفاکس ۶۸ شدیم، به همین دلیل اکسپلویت نهایی ما فقط برای کاربرانی ارائه خواهد شد که از فایرفاکس ۶۸ استفاده می‌کنند. حال اگر ما بخواهیم کاربران وسیعی را مورد حمله قرار بدهیم، باید لیستی از User-Agentهای کاربردی داشته

[۱۶] Selective Poisoning

باشیم که مطمئن شویم اغلب بازدیدکنندگان سایت را با موفقیت هدف قرار خواهیم داد. البته این مسئله به ما این امکان را می‌دهد که حمله خود را بیشتر سفارشی‌سازی و هدفمندتر کنیم.

کاهش تهدید آسیب‌پذیری

در این مقاله به صورت نمادین چند رویکرد ساده برای بهره‌برداری از این آسیب‌پذیری نمایش داده شد و البته رویکردهای پیشرفته‌تر و وکتورهای حمله پیچیده‌تر را در این سند عمومی مستندسازی نکردیم تا امنیت کاربران وسیعی در خطر قرار بگیرد.

با این حال، به منظور اینکه تهدید این آسیب پذیری را کاهش دهیم می توانیم رویکردهای زیر را دنبال کنیم:

۱. **غیرفعال‌سازی کش:** اگر مسئله کش صفحات وب برای سرویس تحت وب اهمیت ندارد، با غیرفعال‌سازی آن می‌توانید این تهدید را از بین ببرید.
۲. **محدودسازی کش:** اگر غیرفعال سازی کش ممکن نیست، کش را به درخواست‌های کاملا استاتیک محدودسازی کنید.
۳. **آزمایش نفوذپذیری وب‌سایت‌ها:** در نهایت برای اینکه متوجه شوید وب‌سایت به ضعف امنیتی مذکور آسیب‌پذیری است یا خیر، باید آن را مورد آزمایش نفوذپذیری قرار بدهید که آزمایشگاه امنیت کی‌پاد آماده انجام هر نوع بررسی و ارزیابی برای مشتریان است.

خلاصه

در این مقاله ما به صورت خیلی ساده به آسیب‌پذیری آلودگی کش وب پرداختیم که درگذشته فرض عموم افراد و توسعه‌دهندگان بر این بود که آسیب‌پذیری مذکور به صورت عملی قابل بهره‌برداری نیست.

در این مقاله به صورت نمادین مشاهده کردیم که این آسیب‌پذیری را می‌توان با تلفیق آسیب‌پذیری‌های دیگر مانند XSS مورد استفاده قرار داد و کاربران وب‌سایت‌ها را مورد نفوذ قرار داد و بر روی سامانه بازدیدکنندگان وب‌سایت آسیب‌پذیر اکسپلویت اجرا کرد، آن‌ها را آلوده به بدافزار کرد و دیگر عملیات‌های مخرب را به واسطه این آسیب‌پذیری بر روی اهداف خود انجام داد. حال برای اینکه تهدید این آسیب‌پذیری را به درستی کاهش داده شود، چندین رویکرد ارائه شده است که در ادامه مقاله به آن‌ها هم اشاره شد.

منابع

- https://www.acunetix.com/blog/articles/what-is-web-cache-poisoning/
- https://www.owasp.org/index.php/Cache_Poisoning
- https://blog.finjan.com/web-cache-poisoning/
- https://www.theregister.co.uk/2018/08/17/web_cache_poisoning/
- https://www.veracode.com/security/cache-poisoning