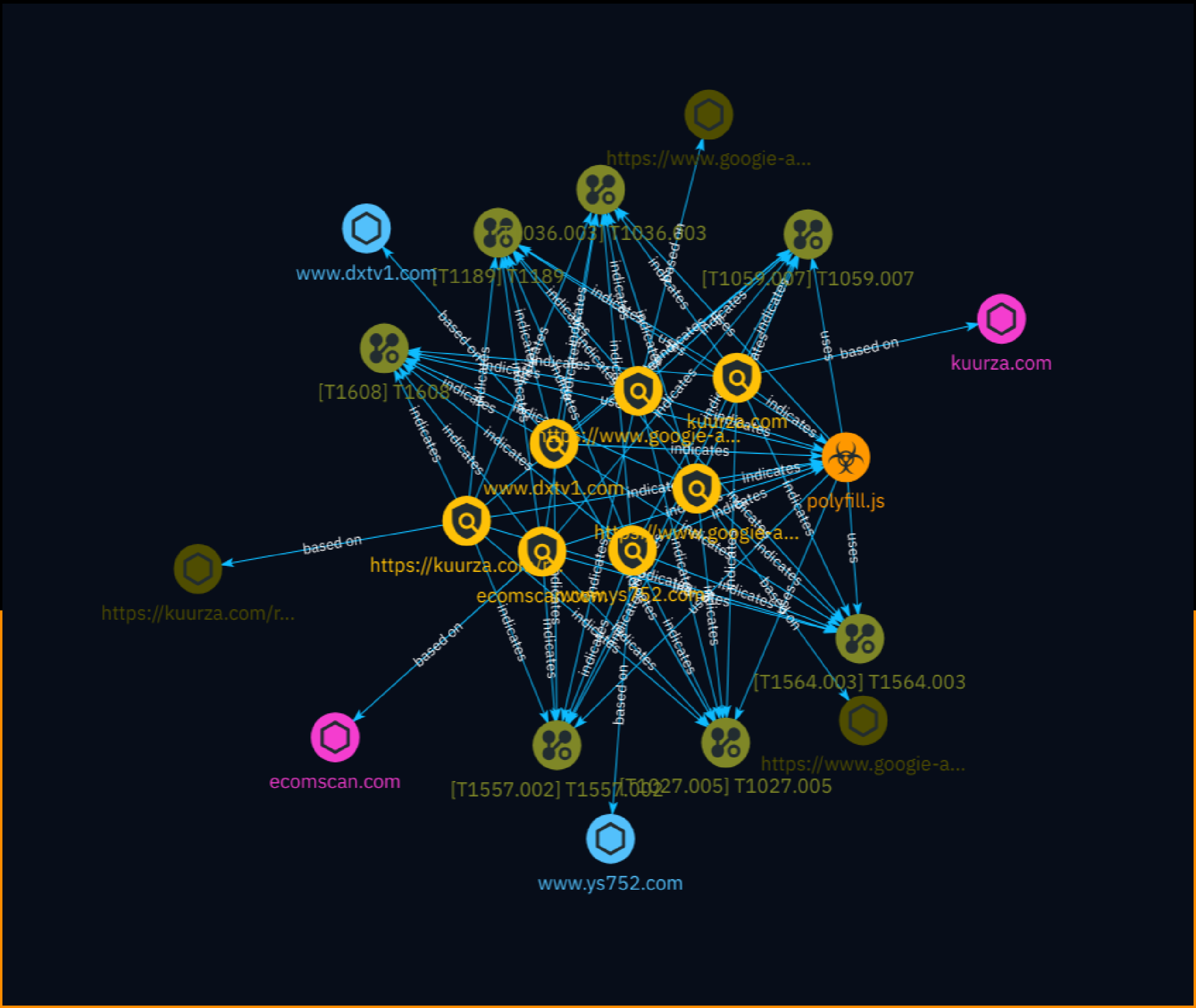
Jun 27 2024

NETMANAGEIT

# Intelligence Report

# Polyfill supply chain attack hits 100K+ sites



# Table of contents

## External References

# Overview

## Description

A malicious Chinese entity acquired control over the popular Polyfill JS open-source project and has been injecting malware into over 100,000 websites that embed the polyfill.io content delivery network. The malware redirects mobile users to a fraudulent sports betting site hosted on a domain impersonating Google Analytics. The attack employs various evasion techniques and targets specific devices and time windows. While trustworthy alternatives are available, it's recommended to remove any references to polyfill.io from your codebase as the library is no longer necessary for modern browsers.

## Confidence

*This value represents the confidence in the correctness of the data contained within this report.*

100 / 100

# Content

N/A

# Attack-Pattern

| Name |
|---|
| T1036.003 |
| **ID** |
| T1036.003 |
| **Description** |
| Adversaries may rename legitimate system utilities to try to evade security mechanisms concerning the usage of those utilities. Security monitoring and control mechanisms may be in place for system utilities adversaries are capable of abusing. (Citation: LOLBAS Main Site) It may be possible to bypass those security mechanisms by renaming the utility prior to utilization (ex: rename `rundll32.exe`). (Citation: Elastic Masquerade Ball) An alternative case occurs when a legitimate utility is copied or moved to a different directory and renamed to avoid detections based on system utilities executing from non-standard paths. (Citation: F-Secure CozyDuke) |

| Name |
|---|
| T1557.002 |
| **ID** |
| T1557.002 |
| **Description** |

Adversaries may poison Address Resolution Protocol (ARP) caches to position themselves between the communication of two or more networked devices. This activity may be used to enable follow-on behaviors such as [Network Sniffing](https://attack.mitre.org/techniques/T1040) or [Transmitted Data Manipulation](https://attack.mitre.org/techniques/T1565/002). The ARP protocol is used to resolve IPv4 addresses to link layer addresses, such as a media access control (MAC) address.(Citation: RFC826 ARP) Devices in a local network segment communicate with each other by using link layer addresses. If a networked device does not have the link layer address of a particular networked device, it may send out a broadcast ARP request to the local network to translate the IP address to a MAC address. The device with the associated IP address directly replies with its MAC address. The networked device that made the ARP request will then use as well as store that information in its ARP cache. An adversary may passively wait for an ARP request to poison the ARP cache of the requesting device. The adversary may reply with their MAC address, thus deceiving the victim by making them believe that they are communicating with the intended networked device. For the adversary to poison the ARP cache, their reply must be faster than the one made by the legitimate IP address owner. Adversaries may also send a gratuitous ARP reply that maliciously announces the ownership of a particular IP address to all the devices in the local network segment. The ARP protocol is stateless and does not require authentication. Therefore, devices may wrongly add or update the MAC address of the IP address in their ARP cache.(Citation: Sans ARP Spoofing Aug 2003)(Citation: Cylance Cleaver) Adversaries may use ARP cache poisoning as a means to intercept network traffic. This activity may be used to collect and/or relay data such as credentials, especially those sent over an insecure, unencrypted protocol.(Citation: Sans ARP Spoofing Aug 2003)

**Name**

T1059.007

**ID**

T1059.007

**Description**

Adversaries may abuse various implementations of JavaScript for execution. JavaScript (JS) is a platform-independent scripting language (compiled just-in-time at runtime) commonly associated with scripts in webpages, though JS can be executed in runtime environments outside the browser.(Citation: NodeJS) JScript is the Microsoft implementation of the same scripting standard. JScript is interpreted via the Windows

Script engine and thus integrated with many components of Windows such as the [Component Object Model](https://attack.mitre.org/techniques/T1559/001) and Internet Explorer HTML Application (HTA) pages.(Citation: JScrip May 2018)(Citation: Microsoft JScript 2007)(Citation: Microsoft Windows Scripts) JavaScript for Automation (JXA) is a macOS scripting language based on JavaScript, included as part of Apple's Open Scripting Architecture (OSA), that was introduced in OSX 10.10. Apple's OSA provides scripting capabilities to control applications, interface with the operating system, and bridge access into the rest of Apple's internal APIs. As of OSX 10.10, OSA only supports two languages, JXA and [AppleScript](https://attack.mitre.org/techniques/T1059/002). Scripts can be executed via the command line utility `osascript`, they can be compiled into applications or script files via `osacompile`, and they can be compiled and executed in memory of other programs by leveraging the OSAKit Framework.(Citation: Apple About Mac Scripting 2016)(Citation: SpecterOps JXA 2020)(Citation: SentinelOne macOS Red Team)(Citation: Red Canary Silver Sparrow Feb2021)(Citation: MDSec macOS JXA and VSCode) Adversaries may abuse various implementations of JavaScript to execute various behaviors. Common uses include hosting malicious scripts on websites as part of a [Drive-by Compromise](https://attack.mitre.org/techniques/T1189) or downloading and executing these script files as secondary payloads. Since these payloads are text-based, it is also very common for adversaries to obfuscate their content as part of [Obfuscated Files or Information](https://attack.mitre.org/techniques/T1027).

**Name**

T1027.005

**ID**

T1027.005

**Description**

Adversaries may remove indicators from tools if they believe their malicious tool was detected, quarantined, or otherwise curtailed. They can modify the tool by removing the indicator and using the updated version that is no longer detected by the target's defensive systems or subsequent targets that may use similar systems. A good example of this is when malware is detected with a file signature and quarantined by anti-virus software. An adversary who can determine that the malware was quarantined because of its file signature may modify the file to explicitly avoid that signature, and then re-use the malware.

| Name |
|---|
| T1189 |

| ID |
|---|
| T1189 |

| Description |
|---|
| Adversaries may gain access to a system through a user visiting a website over the normal course of browsing. With this technique, the user's web browser is typically targeted for exploitation, but adversaries may also use compromised websites for non-exploitation behavior such as acquiring [Application Access Token](https://attack.mitre.org/techniques/T1550/001). Multiple ways of delivering exploit code to a browser exist (i.e., [Drive-by Target](https://attack.mitre.org/techniques/T1608/004)), including: * A legitimate website is compromised where adversaries have injected some form of malicious code such as JavaScript, iFrames, and cross-site scripting * Script files served to a legitimate website from a publicly writeable cloud storage bucket are modified by an adversary * Malicious ads are paid for and served through legitimate ad providers (i.e., [Malvertising](https://attack.mitre.org/techniques/T1583/008)) * Built-in web application interfaces are leveraged for the insertion of any other kind of object that can be used to display web content or contain a script that executes on the visiting client (e.g. forum posts, comments, and other user controllable web content). Often the website used by an adversary is one visited by a specific community, such as government, a particular industry, or region, where the goal is to compromise a specific user or set of users based on a shared interest. This kind of targeted campaign is often referred to a strategic web compromise or watering hole attack. There are several known examples of this occurring.(Citation: Shadowserver Strategic Web Compromise) Typical drive-by compromise process: 1. A user visits a website that is used to host the adversary controlled content. 2. Scripts automatically execute, typically searching versions of the browser and plugins for a potentially vulnerable version. * The user may be required to assist in this process by enabling scripting or active website components and ignoring warning dialog boxes. 3. Upon finding a vulnerable version, exploit code is delivered to the browser. 4. If exploitation is successful, then it will give the adversary code execution on the user's system unless other protections are in place. * In some cases a second visit to the website after the initial scan is required before exploit code is delivered. Unlike [Exploit Public-Facing Application](https://attack.mitre.org/techniques/T1190), the focus of this technique is to exploit software on a client endpoint upon visiting a website. This will commonly give an adversary access to systems on the internal network instead of external systems that may be in a DMZ. Adversaries may also use compromised websites to deliver a user to a malicious |

application designed to [Steal Application Access Token](https://attack.mitre.org/techniques/T1528)s, like OAuth tokens, to gain access to protected applications and information. These malicious applications have been delivered through popups on legitimate websites.(Citation: Volexity OceanLotus Nov 2017)

**Name**

T1564.003

**ID**

T1564.003

**Description**

Adversaries may use hidden windows to conceal malicious activity from the plain sight of users. In some cases, windows that would typically be displayed when an application carries out an operation can be hidden. This may be utilized by system administrators to avoid disrupting user work environments when carrying out administrative tasks. Adversaries may abuse these functionalities to hide otherwise visible windows from users so as not to alert the user to adversary activity on the system.(Citation: Antiquated Mac Malware) On macOS, the configurations for how applications run are listed in property list (plist) files. One of the tags in these files can be `apple.awt.UIElement`, which allows for Java applications to prevent the application's icon from appearing in the Dock. A common use for this is when applications run in the system tray, but don't also want to show up in the Dock. Similarly, on Windows there are a variety of features in scripting languages, such as [PowerShell](https://attack.mitre.org/techniques/T1059/001), Jscript, and [Visual Basic](https://attack.mitre.org/techniques/T1059/005) to make windows hidden. One example of this is `powershell.exe -WindowStyle Hidden`.(Citation: PowerShell About 2019) In addition, Windows supports the `CreateDesktop()` API that can create a hidden desktop window with its own corresponding `explorer.exe` process.(Citation: Hidden VNC)(Citation: Anatomy of an hVNC Attack) All applications running on the hidden desktop window, such as a hidden VNC (hVNC) session,(Citation: Hidden VNC) will be invisible to other desktops windows.

**Name**

T1608

**ID**

T1608

## Description

Adversaries may upload, install, or otherwise set up capabilities that can be used during targeting. To support their operations, an adversary may need to take capabilities they developed ([Develop Capabilities](https://attack.mitre.org/techniques/T1587)) or obtained ([Obtain Capabilities](https://attack.mitre.org/techniques/T1588)) and stage them on infrastructure under their control. These capabilities may be staged on infrastructure that was previously purchased/rented by the adversary ([Acquire Infrastructure](https://attack.mitre.org/techniques/T1583)) or was otherwise compromised by them ([Compromise Infrastructure](https://attack.mitre.org/techniques/T1584)). Capabilities may also be staged on web services, such as GitHub or Pastebin, or on Platform-as-a-Service (PaaS) offerings that enable users to easily provision applications.(Citation: Volexity Ocean Lotus November 2020)(Citation: Dragos Heroku Watering Hole)(Citation: Malwarebytes Heroku Skimmers)(Citation: Netskope GCP Redirection)(Citation: Netskope Cloud Phishing) Staging of capabilities can aid the adversary in a number of initial access and post-compromise behaviors, including (but not limited to): * Staging web resources necessary to conduct [Drive-by Compromise](https://attack.mitre.org/techniques/T1189) when a user browses to a site.(Citation: FireEye CFR Watering Hole 2012)(Citation: Gallagher 2015)(Citation: ATT ScanBox) * Staging web resources for a link target to be used with spearphishing.(Citation: Malwarebytes Silent Librarian October 2020)(Citation: Proofpoint TA407 September 2019) * Uploading malware or tools to a location accessible to a victim network to enable [Ingress Tool Transfer](https://attack.mitre.org/techniques/T1105).(Citation: Volexity Ocean Lotus November 2020) * Installing a previously acquired SSL/TLS certificate to use to encrypt command and control traffic (ex: [Asymmetric Cryptography](https://attack.mitre.org/techniques/T1573/002) with [Web Protocols](https://attack.mitre.org/techniques/T1071/001)).(Citation: DigiCert Install SSL Cert)

# Indicator

| Name |
|---|
| https://www.googie-anaiytics.com/html/checkcachehw.js |
| **Pattern Type** |
| stix |
| **Pattern** |
| [url:value = 'https://www.googie-anaiytics.com/html/checkcachehw.js'] |
| **Name** |
| ecomscan.com |
| **Pattern Type** |
| stix |
| **Pattern** |
| [domain-name:value = 'ecomscan.com'] |
| **Name** |
| www.ys752.com |

| Pattern Type |
|---|
| stix |

| Pattern |
|---|
| [hostname:value = 'www.ys752.com'] |

| Name |
|---|
| kuurza.com |

| Pattern Type |
|---|
| stix |

| Pattern |
|---|
| [domain-name:value = 'kuurza.com'] |

| Name |
|---|
| https://www.googie-anaiytics.com/ga.js |

| Pattern Type |
|---|
| stix |

| Pattern |
|---|
| [url:value = 'https://www.googie-anaiytics.com/ga.js'] |

| Name |
|---|
| www.dxtv1.com |

| **Pattern Type** |
|---|
| stix |
| **Pattern** |
| [hostname:value = 'www.dxtv1.com'] |
| **Name** |
| https://kuurza.com/redirect?from=bitget |
| **Pattern Type** |
| stix |
| **Pattern** |
| [url:value = 'https://kuurza.com/redirect?from=bitget'] |

# Malware

| Name |
|---|
| polyfill.js |

# indicates

| |
|---|
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |

| |
|---|
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |

| |
|---|
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |

| |
|---|
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |

| |
|---|
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |
| |
| **Name** |

| Name |
|---|
| |

| Name |
|---|
| |

| Name |
|---|
| |

## uses

| Name |
|---|
| |

| Name |
|---|
| |

| Name |
|---|
| |

| Name |
|---|
| |

| Name |
|---|
| |

| Name |
|---|
| |

| Name |
|---|
| |

# based-on

| Name |
|---|
|  |
| **Name** |
|  |
| **Name** |
|  |
| **Name** |
|  |
| **Name** |
|  |
| **Name** |
|  |
| **Name** |
|  |

# Domain-Name

| Value |
|---|
| ecomscan.com |
| kuurza.com |

# Hostname

| Value |
|---|
| www.dxtv1.com |
| www.ys752.com |

# Url

| Value |
|---|
| https://kuurza.com/redirect?from=bitget |
| https://www.googie-anaiytics.com/ga.js |
| https://www.googie-anaiytics.com/html/checkcachehw.js |

# External References

- https://sansec.io/research/polyfill-supply-chain-attack
- https://otx.alienvault.com/pulse/667d5bd4a98d33d3486fff19