Office Documents: New Weapons of Cyberwarfare

Jonathan Dechaux dechaux@et.esiea-ouest.fr,
Eric Filiol filiol@esiea.fr,
Jean-Paul Fizaine fizaine@esiea-recherche.eu

Ecole Supérieure en Informatique, Electronique et Automatique
Operational virology and cryptology Lab.
38 rue des docteurs Calmette & Guerin, 53000 Laval France

Cyberwarfare and Cyberweapons

Reallity of cyberwarfare

- August 2007: Espionage case of China against German chancelery.
  - 163 Gb of Gouvernemental data stolen through a Trojan-infected Office document.
- 2009 - 2010: Chinese hackers succeeded in stealing economic and financial data from European Banks, through malicious PDFs.

Document as cyberweapons

- (Open)Office document are good vectors.
- PDF documents are also used nowadays

A Critical Context

The Cyberwarfare picture

- PWN2KILL, May 2010 Paris, challenge has proved the risk is real and high.
  - http://www.esiea-recherche.eu/iawacs2010.html
- Huge technical possibilities on one side, quite no protection and detection capability on the other side.

- Many critical systems are rather secure with a strong security policy enforced.
- Classical approaches are less and less possible, not say impossible.

Context of the Study

Which applications are concerned?

- Office 2010
- OpenOffice 3.x
- All office applications.

The Purpose

- To install malicious payload into the operating system, whithout being detected by any AV.
- We do not want to exploit any vulnerability (target = secure sensitive systems; e.g. combat systems).

MSO: Execution level security settings

Possible level of security

- Level 4 (0x00000004): Disable all macros without notification.
- Level 3 (0x00000002): Disable all macros with notification.
- Level 2 (0x00000003): Disable all macros except digitally signed macros.
- Level 1 (0x00000001): Enable all macros.

Location of settings

- Registery key : HKEY_CURRENT_USER
- \Software\Microsoft\Office\ 12.0\ <Application> \Security
- Application = {Word, Excel, Powerpoint, Access}

MSO: Trusted Location

Definition

Trusted location: A **trusted location** is a directory where macros of documents stored inside are allowed to be executed automatically.

Stored in the registery

- HKEY_CURRENT_USER
- \Software\Microsoft\Office12\12.0\ <Application> \Security\ **Trusted Location**.
- trust value.
- Standalone settings: modifying Word's settings does not affect other Office programs' settings.

OO: Macro Security

Security settings

Both **Macro security level** and **trusted location** are defined in "**Common.xcu**" file at:
Openoffice.org\3\user\registery\data\org\openoffice\Office

Example

```
<node oor:name="Security">
 <node oor:name="Scripting">
  <prop oor:name="MacroSecurityLevel" oor:type="xs:int">
   <value>0</value>
  </prop>
 </node>
</node>
```

OO: Trusted Location

Example

Set the root directory as Trusted location

```
<node oor:name="Security">
 <node oor:name="Scripting">
  <prop oor:name="SecureURL" oor:type="oor:string-list">
   <value>file:///C:/</value>
  </prop>
 </node>
</node>
```

The use of 'AutoExec' event with MSO

The fact

Able to naturally bypass the level 2 of execution.

- Several events are available: AutoNew, Open, Close, Exit, Exec
- Applied on template named *Normal.dotm* and stored inside MSO's users settings file.
- Execute the macro at opening event even if any macro are not allowed to be executed (Level 2).

MSO & OO.ORG: The integration

MSO&OO.ORG are both:

Based on the W3C specification. But the integration is totally different.

MSO's integration

- Office makes it easier to create signatures.
- It is possible to create self-signed certificates.
- They are stored inside _*rel\.rel* file whithin the document.

Openoffice's integration: X509Certificate

No significant change about signature since 2006, the first study. Black Hat 2009, Amstersdam, E.Filiol J.-P. Fizaine, Openoffice v3.x Security Design Weaknesses.

MSO case
Change to the lowest level: 0

Interessting Keys: HKEY_CURRENT_USER

Path: Software\\\\Microsoft\\\\Office\\\\12.0\\\\Word\\\\Security

Windows API: RegOpenKeyEx, RegSetValueEx, RegCreateKeyEx, RegCloseKey

Example

```
RegOpenKeyEx(HKEY_CURRENT_USER, path, 0,
        KEY_ALL_ACCESS, &hkey);
RegSetValueEx(hKey, warning, 0, REG_WORD,
        (const BYTE*)nNumber, sizeof(number));
RegClose(hkey);
```

MSO case

Set the directory *c:\Users* as a Trusted Location.

KEY:	HKEY_CURRENT_USER
Path:	Software\\Microsoft\\Office\\12.0\\Word\\Security\\Trusted\\Locations
Path2:	Software\\Microsoft\\Office\\12.0\\Word\\Security\\Trusted\\Locations\\Location3

Example

```
RegOpenKeyEx(HKEY_CURRENT_USER, path, 0,
        KEY_ALL_ACCESS, &hkey);
RegCreateKeyEx(hkey,location, 0, NULL,
        REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS,
        NULL, &hkey2, NULL)
RegOpenKeyEx(HKEY_CURRENT_USER, path2, 0,
        KEY_ALL_ACCESS, &hkey2);
```

MSO case

Set the directory *c:\Users* as a Trusted Location.

Example

```
RegSetValueEx(hKey2, description, 0, REG_SZ,
        (const BYTE*)("1"), 32);
RegSetValueEx(hKey2, path_t, 0, REG_SZ,
        (const BYTE*)TEXT("C:\\Users\\"), 32);
RegSetValueEx(hKey2, allow, 0, REG_DWORD,
        (const BYTE*)&number, sizeof(number));
RegCloseKey(hkey2);
RegCloseKey(hkey);
```

OO.ORG case

Change the Macro security level to the lowest: 0

Where are the settings ?

Settings are stored in only one file! No use of specific library is needed, the C Standard Library is sufficient.

The Algorithm: based on standard I/O function

1. Forge the Path.
2. Locate the position inside the file.
3. Insert the value:
   <node oor:name="Security"> <node oor:name="Scripting"> <prop oor:name="MacroSecurityLevel" oor:type="xs:int"> <value>0</value> </prop> </node> </node>
4. Update by restart the application.

OO.ORG case
Trusted Locations

The Algorithm

It is exactly the same algorithm that manages the security level.

1. Forge the Path.
2. Locate the position inside the file.
3. Insert the value:
   <node oor:name="Security"> <node oor:name="Scripting"> <prop oor:name="SecureURL" oor:type="oor:string-list"> <value>file:///C:/</value> </prop> </node> </node>
4. Update by restarting the application.

introduction to k-ary malware

k-ary malware definition

Malware made of k different, innocent-looking (from the AV point of view). Each of them can (inter)act independently or not and can either be executed in parallel or in sequential. Not all the parts are necessarily executable. The cumulative action of each part defines the malware action.

Proof of concept

- E. Filiol, Journal in Computer Virology, 2007.
- Hack.lu 2009, A. Desnos, Implementation of K-ary viruses in Python.

Two waves of attack: the use of 2-ary malware

The fact

Suppose the security level is set to the paranoid mode, it is impossible to change the level from inside the macro. (Journal in Computer Virology, 2006, D. de Drézigué, J.- P. Fizaine, N. Hansma, In-depth Analysis of the Viral Threats with OpenOffice.org Documents).

The algorithm

- External change of settings by a program.
- Macro performs its payload.
- Demos
  1. How to install in Trojan.
  2. How to perfom a simple DoS.

Why this approach?

The fact

Attacking (secure) systems becomes really complex. Just exploiting one or more vulnerability does no longer suffice. Installing a functionnally sophisticated program is less and less easy. The solution is to split the viral information into many pieces!

- Real case: secure systems generally filter and forbid packed binaries/shellcodes.
- Using 2-ary malware is a powerful alternative.
- The first executable performs a innocent, generally legitimate simple action.
    - Equivalent to the packing step/functionality.
- The office document then installs more complex malware transparently and silently.

General Algorithm

The same attack exists for Openoffice: Black hat 2009, Amsterdam, E.Filiol, j.-P. Fizaine, OpenOffice v3.x Security Design Weaknesses.

Once apon a time, Alice and Bod are working for the Dept. of Industry. Alice sent a sensitive document to her collegue Bob. But Charly a spy is very interested in Alice's work...

1. Get the signature data.
2. Signature Information removal.
3. Forge a new certificate.

Get & remove the signature

Collect the following informations:

Town, compagny, email, address and name by analyzing the certificate.

Remove from _*rel\\.rels*_

```
<Relationship Id="rId4" Type="http://schemas.
openxmlformats.org/package/2006/relationships/
digital-signature/origin" Target="_xmlsignatures/
origin.sigs"/>
```

Get & remove the signature

Remove from *Content_Types.xml*

```
<Default Extension="sigs" ContentType="application/
vnd.openxmlformats-package.digital-signature-origin"/>

<Override PartName="/_xmlsignatures/sigs1.xml"
ContentType="application/vnd.openxmlformats-package.
digital-signature-xmlsignature+xml"/>
```

Refinement

Refine the time modification and the author's name.

Protection and Counter-measures

1. Use of **Public Key Infrastructure**
2. whenever self-signed certificates are used:
   Check the **serial number, timestamp** and validity systematically.
   The serial number is supposed to be **unique**.

Trusted documents

Location of settings

- Activate properties for only one document without modifying the security.
- Registery key : HKEY_CURRENT_USER
- \ Software\ Microsoft\ Office\ 14.0\ <Application> \Security\ Trusted Documents\ TrustRecords
- *Application* = {Word, Excel, Powerpoint, Access}
- Binary name = Document path.
- Binary value = Document creation date and document last opening.

Demonstration

Mobile storage attack with a 2-ary malware:

- One Word file protected with a password
- One executable program to get the password

Conclusion

(Open)office are efficient cyberweapons

- Techniques can be implemented with simple, reduced size code, and be very powerful.
- Antivirus software fail to detect those attacks systematially, even KAV.
- (Open)Office documents intensively exchanged.
- Security no longer relies on the application only (look at the OS levelas well).
- Futher works:
  - PDF version.
  - Anti-forensics,
  - More execution environments.
  - More powerful cyberweapons.

Thanks for your attention.
Questions?