ExeFilter

An open-source framework for active content filtering

CanSecWest 2008 – 28/03/2008 – http://cansecwest.com

Philippe Lagadec – NATO/NC3A
philippe.lagadec(à)nc3a.nato.int

ExeFilter Goals

- To protect *sensitive networks* against attacks involving **files**, **e-mails** and **active content**.

- To ensure that **only known and controlled file formats** enter the system.
- To filter all unwanted **active content** from **external sources**.

Threats

- Many common file formats and attachments may contain hidden malicious content:
  - **Macros** or **OLE objects** in MS Office documents (Word, Excel, …), OpenDocument, RTF
  - **Scripts** in HTML pages, PDF, XML, Flash
  - **Executable files**
  - **Exploits** in malformed documents and files
- Once a user launches untrusted active content on his workstation, it may be compromised.
- Antivirus software cannot catch all attacks.

Observations

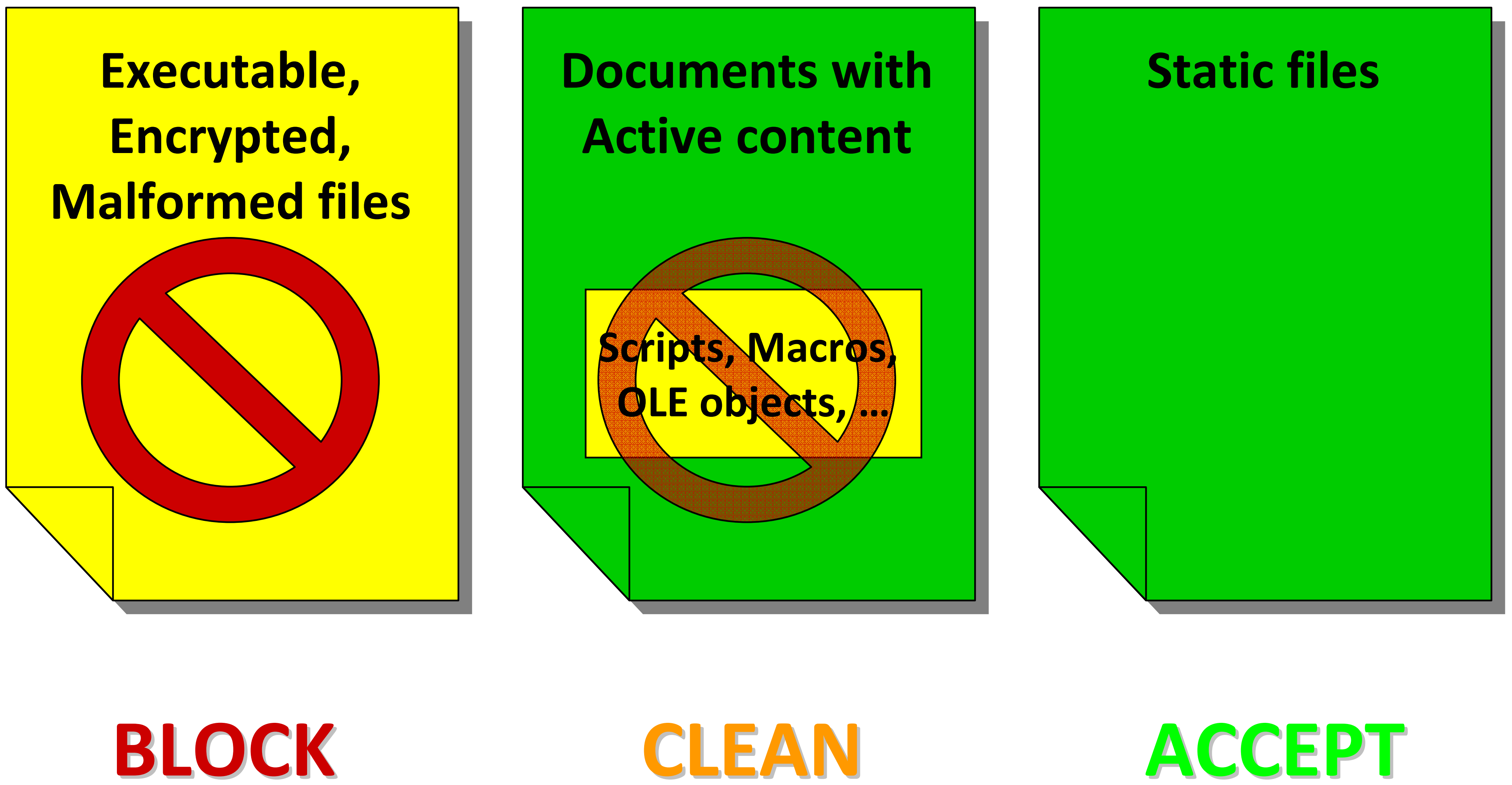
- **On sensitive systems, almost all files received or downloaded from *external sources* do not need active content.**

- Most security products focus on **antivirus** and **antispam** features.
- They usually provide little help against **targeted attacks using active content**.
  - E.g. they can usually be bypassed.

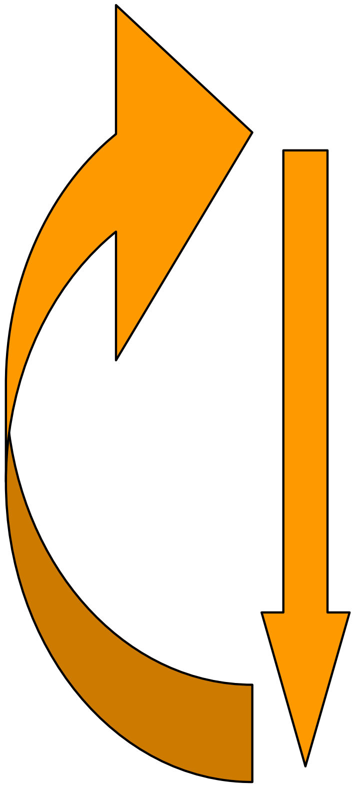
ExeFilter project

- Developed by DGA/CELAR (French MoD) since 2004, written in Python.
- Released as open-source in 2008
    - CeCILL license, GPL-compatible
    - Core maintainers: DGA/CELAR and NATO/NC3A
- Generic engine to filter files according to a white-list policy
    - Only chosen and known formats are allowed
- Extensible for various needs:
    - Filter on **gateway**: Web, E-mail, File transfer, Diode, …
    - Filter on **workstation**: Removable devices

ExeFilter policy


Executable, Encrypted, Malformed files
Documents with Active content
Scripts, Macros, OLE objects, …
Static files
BLOCK
CLEAN
ACCEPT

ExeFilter – principle



- **Each file is analyzed:**
  - **Format detection** according to file **name** AND **content**.
  - **BLOCKED** if format is not explicitly allowed by policy.
    - Executable files, scripts, unknown formats, encrypted, malformed, …
  - **CLEANED** if active content
    - Macros in Office, scripts in HTML, …
  - **ACCEPTED** if harmless
    - Simple text, bitmap pictures, …
  - **Antivirus scan** (to detect known exploits)
  - **Recursive analysis** if container (Zip archives)

Supported formats (v1.1.0)

Formats	Active content	Default Action
Text	-	**Accept**
JPEG,PNG,GIF,BMP	-	**Accept**
Audio: MP3, WAV	-	**Accept**
Video: AVI	-	**Accept**
HTML	Scripts, Objects	**Clean**
PDF	JavaScript, Embedded files	**Clean**
Word, Excel, PPT	Macros, OLE Objects	**Clean**
RTF	OLE Objects	**Clean**
Zip archive	Compressed files	**Clean**
Any format	unknown, malformed, encrypted	**Block**

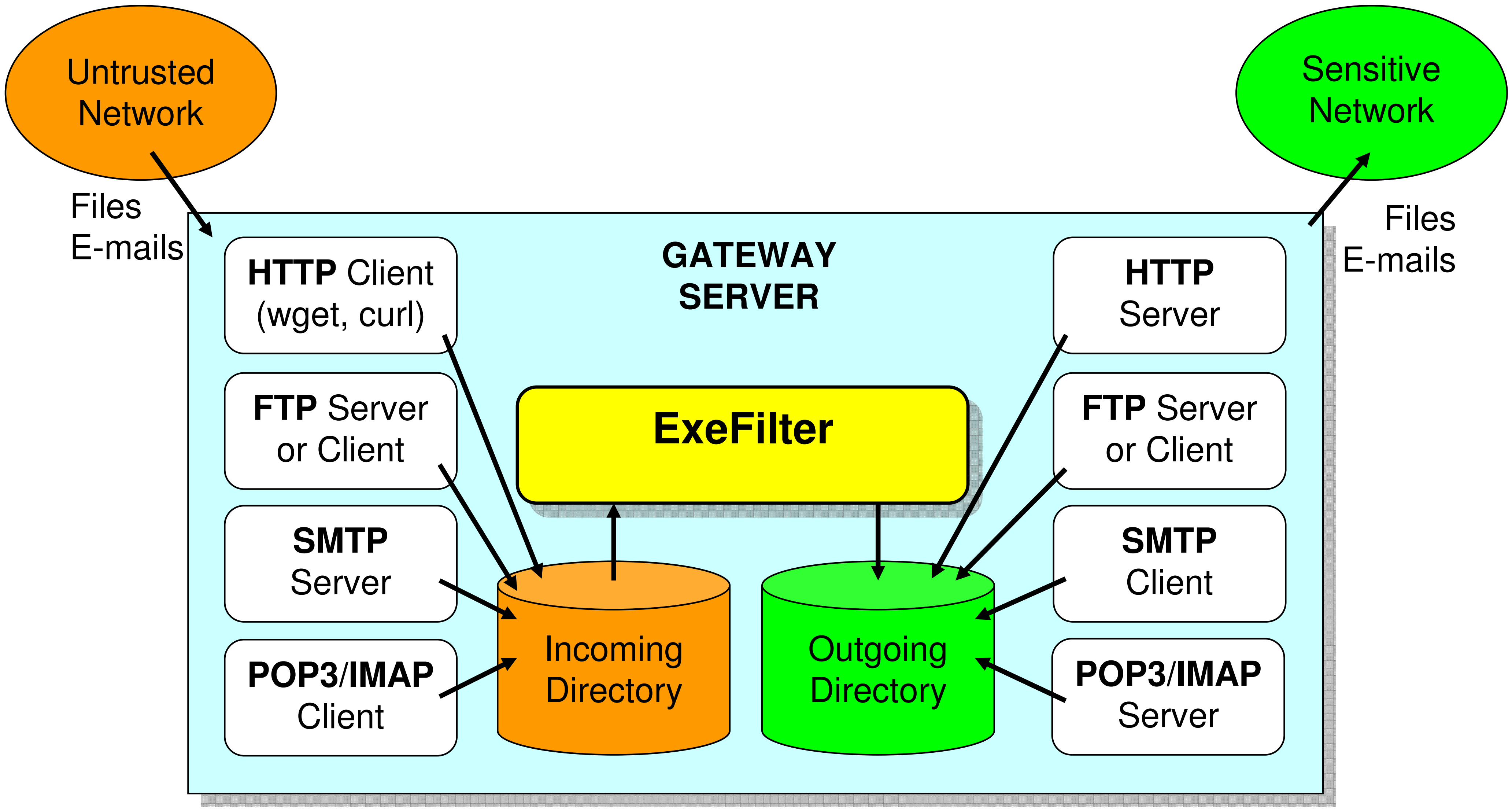
How to use ExeFilter

- **As a script** from shell / cmd.exe:
  - Takes a list of dirs/files as input, filters them and copy the cleaned result in a destination dir.
  - **ExeFilter.py** <source dirs/files> **-d** <dest dir>
  - Optional config file for parameters and policy

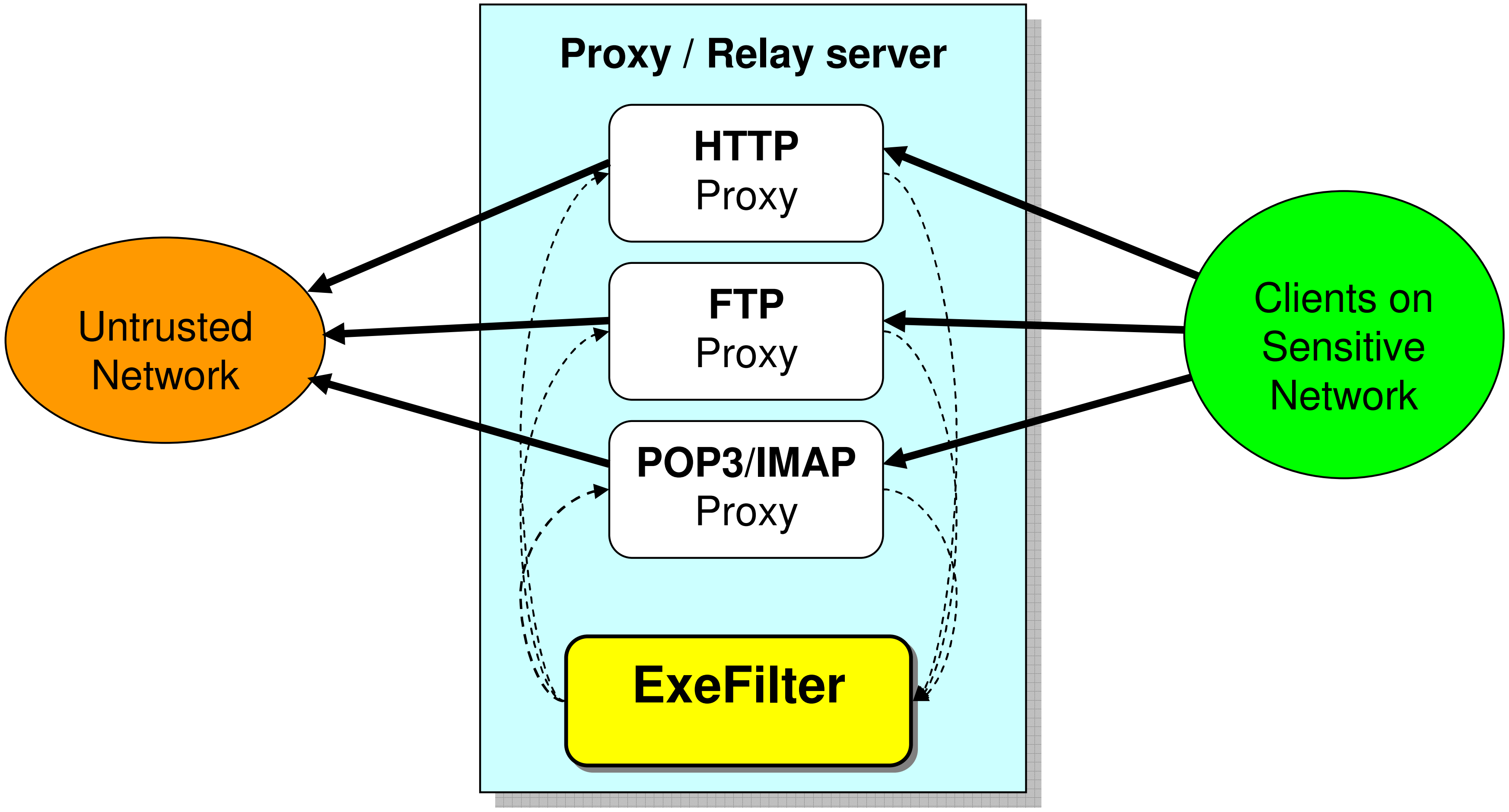
- **As a Python module** from another Python script:
  - Import ExeFilter
  - Read the fine manual and example code in ExeFilter.py

demo

ExeFilter in a gateway server



ExeFilter in a Proxy / Relay



ExeFilter vs. similar products

- **Advantages**:
  - **White-list formats filtering,** whereas most products are designed for Black-list
  - **Strict matching of file extensions and content**
  - Focused on **active content removal** and **targeted attacks**
  - Viruses, worms and Trojan horses usually rely on active content: **most of them are removed "by design"**
  - Generic filtering engine, **Open-source**, Modular
  - **Extensible**: any file processing may be added
- **Constraints**:
  - Restrictive for users: no active content, no unknown formats (except specific configuration)
    - Mostly suitable for *sensitive* networks with a strict policy
  - Most websites require scripts in HTML and/or Flash
  - Detecting all unknown exploits is obviously impossible
  - Need to update filters when formats change

Why strict matching of file extension/content is necessary

- Most Operating Systems rely on **file extension** to choose the application to open a file (in absence of metadata)
- However, some security products only rely on **file header**:
  - **PDF filter bypass**:
    - PDF is usually expected to start with **"%PDF-1.x" header**
    - But PDF specs allow to insert 1000 chars before header
    - If you insert "<HTML>" before, some products treat the file as HTML instead of PDF
    - Example file: "Fake_HTML_header**.pdf**":

```
<HTML>%PDF-1.4 …
```

  - **HTML filter bypass**:
    - Browsers allow any text as HTML, **"<HTML>"** is not even required.
    - If you insert a simple PDF header before HTML content, some products treat it as PDF, and HTML content is not sanitized…
    - Example file: "Fake_PDF_header**.html**":

```
%PDF-1.4 <HTML><SCRIPT>…</SCRIPT></HTML>
```

Focus: MS Office filter

- **1) Check format**:
  - MAGIC = D0 CF 11 E0 A1 B1 1A E1
  - Parse OLE2 structure with OleFileIO (improved version from PIL)
- **2) Detect encrypted files**: OLE properties
- **3) Detect OLE Package objects**
- **4) Detect and disable macros**:
  - Word: “Macros” storage / Excel: “_VBA_PROJECT_CUR”
  - Remove with Win32 API or pattern replace
  - Or use F-Prot antivirus macros removal feature (better but much slower)

Detecting exploits in documents

- Exploits in files are usually found in malformed documents.
  - MS Office, JPEG, GIF, …
- **Techniques used in current ExeFilter version:**
  - **1) Antivirus scanning** to detect known exploits
    - But not 100% accurate in practice, even for widespread exploits
  - **2) Format parsing** to detect malformed files
    - …As long as the parser itself is not vulnerable.
    - Trade-off between parser complexity and detection accuracy
- **Techniques to be investigated in future versions:**
  - **3) Specific scanners** to improve detection of known exploits
    - Examples: SourceFire OfficeCat for MS Office, SecureWorks Fess
  - **4) File carving**, look for suspicious patterns
    - Example: detect PE header inside a MS Office document, detect NOPs
  - **5) Heuristic techniques**: entropy to detect encryption/compression, …

Focus: HTML Filter

- **1) Check format**: Python HTMLParser
- Several improvements were needed to **avoid obfuscations**:
  - Remove null bytes before parsing, as IE does
  - Proper Encoding support: Unicode, UTF-8, …
    - Check BOM and META tags
  - Improve character entities support in attributes
    - Example: `<a href="ja&#118;ascript:…">`
- **2) Remove active content**:
  - Tags: SCRIPT, OBJECT, EMBED, APPLET, IFRAME, …
  - Script attributes: onLoad, onMouseOver, ...
  - Script URLs: "javascript:…", "vbscript:…", …

- However, this is mostly useful for HTML files/e-mails, not for "live" web pages

How to filter live web pages

- **Issue**: most websites rely on JavaScript and/or Flash, Java applets, etc
  - Filtering all active content in HTML is not practical
  - Allowing it is not secure
- **Current version:** allow scripts when filtering HTTP
- **Solutions to be investigated in future versions:**
  - **Solution 1**: White list of trusted sites/domains
    - Example: NoScript plugin for Firefox, IE zones
    - Heavy to manage for administrators, risky if users manage the list.
  - **Solution 2**: Analyze script code, only allow safe features
    - Need a parser for each language (ex: SpiderMonkey for js)
    - May be complex and prone to vulnerabilities

Development status

- Current version 1.1.x is still under active development
  - This is a prototype, not for production use
- Many features to add, test and fix
- Issues to fix on non-Windows OSes
- Ongoing translation to English (gettext)
- All source code is obfuscated in French ☺ (API and comments)

Planned ExeFilter evolutions

- **New file formats** : OpenDocument, Open XML, …
- Simple **GUI**
- **Clamd-like daemon interface** for easy integration
- **ICAP server** to improve integration with various HTTP proxies (Squid3, Webwasher, BlueCoat, …)
- **MIME support** to provide e-mail relay integration
- **Improved portability** on Linux, BSD, MacOSX
- Support for more **antivirus engines**
- “Scan only” mode
- Filters for outgoing files (clean metadata, block confidential files, …)

How to Contribute

- Project website:
  - http://admisource.gouv.fr/projects/exefilter
- Test ExeFilter on your platforms and report bugs, comments, new ideas, …
- Test its security, Fuzzing, Report vulnerabilities
- Develop new filters (please contact us before)
- Provide support for proxies/relays/antivirus integration
- Help improve docs, source code, translation

Conclusion

- ExeFilter is a new open-source framework to improve protection of sensitive networks against targeted attacks using files and e-mails.
- All contributions will be helpful to improve it.
  - Tests, new filters, developers, ideas
- Please visit the project website:
  - http://admisource.gouv.fr/projects/exefilter

Any questions ?

```
if question:
    try:
        answer()
    except DontKnowError:
        next()
```

Links

- ExeFilter:
  - http://admisource.gouv.fr/projects/exefilter
- File formats:
  - http://www.wotsit.org/
  - http://hachoir.org/wiki/FileFormatResources
- File formats and Malware (French):
  - http://actes.sstic.org/SSTIC03/Formats_de_fichiers/
- Sourcefire OfficeCat:
  - http://www.snort.org/vrt/tools/officecat.html
- SecureWorks Fess:
  - http://www.secureworks.com/research/tools/fess.html
- OleFileIO:
  - http://www.decalage.info/python/olefileio
- NoScript plugin for Firefox:
  - http://noscript.net/
- SpiderMonkey Javascript engine:
  - http://www.mozilla.org/js/spidermonkey/

How it works

- A **Filter** for each file format:
  - List of allowed extensions
    - E.g. for MS Word: .doc, .dot
  - Format compliance checks
  - Active content removal, if applicable
- For each file, all relevant filters are called according to the file extension.
  - Example: .DOC → Word, RTF and Text filters.
- For each **Container** format (e.g. Zip), all embedded files are extracted and analyzed.

New formats

Formats	Active content	Default Action
Unicode Text	-	**Accept**
XML (known schemas)	-	**Clean**
OpenDocument (OpenOffice, StarOffice)	Macros, Scripts, OLE Objects	**Clean**
Open XML (MS Office 2007)	Macros, OLE Objects	**Clean**
Archives: TAR, GZ, RAR	Compressed files	**Clean**
MIME (e-mails), MHT	Attachments	**Clean**

Configuration / Policy

- One or several configuration files:
  - “ini” syntax
  - General parameters
  - Parameters for each file format:
    - Allowed or not
    - Specific parameters (which active content to remove)

Design

- Python:
  - high-level source code, portable, robust
  - Only call external tools when needed
- Object-oriented, modular
- Performance: files are read only when needed
- Supported antiviruses: ClamAV (clamd), F-Prot
- Current version 1.1.0:
  - Windows XP/2003: OK
  - Linux, MacOSX: partial support, known issues with encoding