Security Analysis of eIDAS – The Cross-Country Authentication Scheme in Europe

Nils Engelbertz, Nurullah Erinola, David Herring,
Juraj Somorovsky, Vladislav Mladenov, Jörg Schwenk
Chair for Network and Data Security,
Horst Görtz Institute for IT-Security, Ruhr University Bochum

Abstract

In 2014, the European Commission released the eIDAS regulation to target the compatibility of cross-country electronic services within the European Union. eIDAS (electronic IDentification, Authentication, and Trust Services) defines implementation standards and technologies for electronic signatures, digital certificates, Single Sign-On (SSO), and trust services. It is based on well-established standards, such as SAML, to achieve high security and compatibility between EU countries.

In this paper, we present the first security study of authentication schemes used in eID services. Our security analysis shows that 7 of the 15 European eID services were vulnerable to XML-based attacks which enabled efficient Denial-of-Service (DoS) and Server Side Request Forgery (SSRF) attacks. On 5 of the 15 eID services, we were even able to exfiltrate locally stored files and send these files to an arbitrary domain. To support the developers and security teams of eID services, we implemented a Burp Suite extension to execute fully-automated or semi-automated tests. Additionally, we summarize best practices related to eID-based authentication and SSO in general.

1 Introduction

In the last few years, European countries have worked on developing strong authentication schemes based on electronic identification (eID) cards. The main goal of these authentication schemes is to provide access to different services, called eID services. For these services, access should be provided for citizens and organizations by using information already available on eID cards, for example, on the personal ID card which was issued by a government institution. Many countries developed their own authentication schemes based on well-established technologies that provide functionalities for secure browsing, login mechanisms, SSO, or exchanging confidential data over untrusted networks. In order to meet national requirements and regulations, countries chose differing technologies and implemented specific extensions and functionalities. This caused incompatibilities between eID services and European cross-country authentication became impossible.

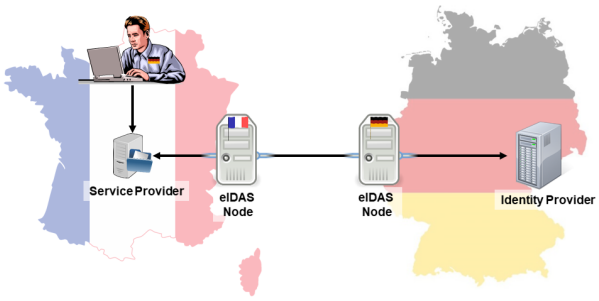

Figure 1: Overview of the eID/eIDAS ecosystem depicting the End-User, Service Provider (SP), Identity Provider (IdP), and both eIDAS Nodes.

eIDAS. In 2014, the European Commission addressed these cross-country incompatibilities for eID authentication by releasing the eIDAS regulation and defining different communication profiles used for authentication and trust establishment. Throughout this paper, we refer to this authentication scheme as *eID/eIDAS ecosystem*. The scheme uses SAML [12] as an authentication protocol.

Figure 1 gives a high level overview of cross-country eIDAS authentication. In our example, a German End-User wishes to use services provided by a French service provider (SP). The End-User cannot simply use the German IdP directly to authenticate at the French SP since the SP could not process the authentication token issued by the IdP. Therefore, eIDAS Nodes are used to translate the tokens and make them compatible between eID schemes. The SP first forwards the request to the French eIDAS Node, which translates the request for

the German eIDAS Node. The German eIDAS Node then forwards the compatible request to the authoritative IdP, which can issue authentication tokens for the End-User. The authentication token is finally translated for the French SP using eIDAS Nodes.

Security of eIDAS. In recent years, it has been shown how to break SAML-based SSO systems and login as an arbitrary user [62, 64, 37], read arbitrary files from SAML servers [37], or how to break XML Encryption and decrypt the exchanged SAML assertions [29, 26, 27, 62]. Because eIDAS makes use of these technologies, such attacks present a serious threat to the eID users and their prevention is, therefore, of high importance.

Automated Security Analysis. In order to support eID developers in their development process, we extended the tool EsPReSSO (Extension for Processing and Recognition of Single Sign-On Protocols), which facilitates in analyzing different SSO protocols and their used information flow. We implemented a prototype of the attacks described in Section 3 into EsPReSSO so that eID developers are able to search for known vulnerabilities.

Discovered Vulnerabilities. The relevance of these vulnerabilities is proven in our evaluation, as we revealed security flaws in 7 of 15 eID services which enabled attacks such as DoS, SSRF, and on 5 of 15 systems, even unauthorized file exfiltration. We reported the discovered vulnerabilities to the affected providers and national CERTs. We also cooperated with the system developers on implementing the countermeasures and provided a second test to verify the implemented fixes.

Contributions. The contributions of this paper can be summarized as follows:

- We present the first security evaluation of existing eID services and reveal security gaps in 7 of the 15 eID services.
- We provide a comprehensive overview of the attacks relevant to eID scenarios. These attacks target the underlying TLS protocol [14], XML parser (XXE attacks) [70], and cryptographic standards like XML Signature [25] and XML Encryption [15].
- We provide a tool to facilitate the security analysis of eID services, supporting developers and security experts to discover security flaws.
- We summarize the lessons learned in security guidelines and a Best Current Practices section for the deployment of secure eID infrastructures.

2 Foundations

This section summarizes relevant foundations regarding SSO, SAML, and the utilization of SAML in eIDAS.

2.1 Single Sign-On (SSO)

Single Sign-On (SSO) is a concept to log a user into a Service Provider (SP) without storing any credentials on that SP. For this purpose, SSO uses a trusted third party called Identity Provider (IdP). An abstract overview of SSO is depicted in Figure 2.

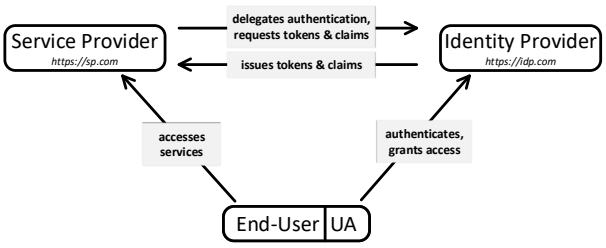

Figure 2: An abstract overview of the entities in SSO and their relation to each other.

The IdP manages user identities and provides at least one authentication mechanism, e.g., username/password. The IdP issues authentication tokens containing information about the authenticated End-User. The End-User is a human being navigating a user agent (UA), e.g., a web browser. The SP is a publicly available service offering access to resources for registered End-Users. If the End-User visits the SP and requests access to restricted resources, authentication is required. The SP delegates authentication of End-Users by issuing an *Authentication Request* to an IdP. Afterwards, the SP verifies the authentication tokens generated by an IdP to authenticate the End-User.

SAML. SAML is an XML [9] standard for exchanging authentication and authorization statements about *subjects* [12],and uses XML-based assertions to transmit these statements. A SAML assertion contains several essential components: The `Issuer` element specifies the SAML authority that is making the claim(s) in the assertion – the IdP. The assertion's `Subject` defines the principal about whom all statements within the assertion are made. Further elements are included to specify message validity or user-defined statements relevant for the message context.

To protect the integrity of the security claims made by the Issuer, the whole `Assertion` element must be protected with a digital signature following the XML Signature specification [16]. Usage of the SAML assertions in various XML messages is described in the SAML Bindings specification [60].

2.2 eIDAS Services

Many EU countries defined and implemented their own singular eID scheme. Differences between the schemes

prevent member states from seamlessly exchanging electronic identification data and trust services.

In order to achieve compatibility in cross-boarder eID communication, eIDAS was defined [54]. eIDAS does not provide a standalone SSO solution, but rather specifies a SAML 2.0 based compatibility layer between different eID implementations [19, 12]. The components facilitating this cross-border information exchange are called *eIDAS-Nodes* [19]. As depicted in Figure 1, eIDAS Nodes "translate" incoming SAML requests from the SP into authentication requests compliant with the eID scheme of the end user's country of origin. On the other side of this process, the eIDAS-Nodes also convert the authentication tokens generated by one country into tokens which can be processed by the SP.

3 XML Attacks on eIDAS Services

eIDAS is based on SAML, which is, in turn, based on eXtended Markup Language (XML). Consequentially, eIDAS services may be susceptible to XML-based attacks. In this section, we focus on Document Type Definition (DTD) attacks. After sketching out our attacker model, we provide an overview of common goals for DTD attacks and appropriate attack vectors.

3.1 Attacker Model and Prerequisites

We consider a Web-Attacker that is capable of generating XML messages, crafting and sending requests, and selecting the appropriate encoding for each part of the submitted request. Furthermore, the attacker controls a publicly available server (henceforth *attacker-listener*) and can observe requests made to this server with arbitrary protocols. As we focus on DTD and XML External Entity attacks, no third party is involved. The goals of the attacker can roughly be categorized into DoS, SSRF, and File Exfiltration, and are further described in the following sections.

3.2 Denial-of-Service (DoS)

DoS attacks aim at decreasing the availability of the service under attack. This is primarily achieved by making the target consume large amounts of computational resources such as memory, bandwidth, or processing power while only investing a fraction of the resources by the attacker [69].

Billion Laughs Attack. A well-known DTD-based DoS attack is the so-called *Billion Laughs Attack* [32, 66]. This attack employs recursively defined *Internal General Entities*, forcing the XML parser to expand a relatively small input document into a document which can reach several gigabytes in size [69, 66]. An example is given in Listing 1. First, an entity `ent0` is declared, referencing the string *DoS*. Next, an entity `ent1` references a concatenation of several times `ent0`. This back-referencing to multiple instances of entities, which is defined in the preceding step, is repeated and results in an exponential expansion of the document size.

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

Listing 1: The *Billion Laughs Attack* abuses limited recursion of General Entities to exponentially expand the document size.

DoS Using External Entities. If the XML parser resolves External Entities, a plethora of DoS attack-vectors may be available. For example, an adversary might be able to induce network requests for large remote files, thereby exhausting network or memory capacity of the system under attack. By inducing a large number of outgoing requests, resources of both the targeted XML parser and the destination of the forged requests may quickly become exhausted [70]. An overview regarding how such requests can be forged is given in the next sections.

Other Techniques. Several variants of the above attacks exist. The *Quadratic Blowup* attack declares a single XML Entity that expands into a large string of several megabytes, greatly exceeding the size of the document. The *Recursive Entity* attack exploits XML parsers by recursively resolving nested entities [66].

3.3 SSRF

One considerable area of attack exposed by many XML parsers is their capability to deal with various URL handlers. The following paragraphs explore some methods on how this can be abused for Server Side Request Forgery (SSRF).

SSRF Using External DTD. A simple way to force a vulnerable XML parser to perform an outgoing request is to reference an external DTD. The example in Listing 2 shows how the XML parser can be induced to query a service in its local network, which would otherwise be unreachable for an external adversary.

```
<!DOCTYPE data SYSTEM "http://192.168.178.2/shutdown">
<data>arbitrary content</data>
```

Listing 2: Using external DTDs to induce server side requests [66].

An alternative for the SYSTEM keyword is PUBLIC "id" as shown Listing 3.

```
<!DOCTYPE data PUBLIC "id" "http://192.168.178.2/shutdown">
<data>arbitrary content</data>
```

Listing 3: The PUBLIC keyword references an external DTD associated with an identifier *id*.

SSRF Using External (Parameter) Entities. In addition to external DTDs, External Entities can be used to cause server side requests, as shown in Listing 4. Because some parsers are able to restrict the allowed protocol handlers, it may be beneficial to try a number of different protocols besides http. Examples include, but are not limited to, ftp://, smb://, http://, https://, file://, and the short UNC path form //.

```
<!DOCTYPE data [
  <!ENTITY extEnt SYSTEM "http://192.168.178.2/shutdown">
]>
<data>&extEnt;</data>
```

Listing 4: Causing Server Side Requests using External Entities

Another way to force the parser to perform outgoing requests is to use Parameter Entities [66].

Other Techniques. The XML specification provides additional methods that can be abused to forge Server Side Requests from the XML parser. Most prominently, schemaLocation and noNamespaceSchemaLocation can cause insecurely configured parsers to issue network requests [66]. The xInclude extension [38] provides the opportunity for an additional attack.

3.4 File Exfiltration

Exfiltration of file content from the parser's local file system can be feasible if a direct feedback channel at the application level exists. Exfiltration is also possible if file content can be included in forged requests to a destination under the adversary's control.

File Exfiltration Using Direct Feedback. In contrast to the External Entity's intended purpose of including additional DTDs from external sources, an adversary can abuse the functionality of External Entities to include files that are otherwise inaccessible. In Listing 5, we assume a direct feedback channel, e.g., the application returns the contents of the <data> element in its response.

```
<!DOCTYPE data [
  <!ENTITY extEnt SYSTEM "file:///etc/passwd">
]>
<data>&extEnt;</data>
```

Listing 5: Using a Direct Feedback channel of the <data> element to read out /etc/passwd

If the content of a referenced file is not well-formed according to the XML specification, well-behaved parsers will abort entity expansion with an exception. Several techniques are known to circumvent this restriction using Parameter Entities to wrap file contents in a <![CDATA []]> block [70, 66].

File Exfiltration Using SSRF. Even if no direct feedback channel is available, file contents can sometimes still be extracted. If the parser can establish network connections, an adversary can proceed as follows: an external DTD can be used to declare Parameter Entities, which can then be included in URLs of forged parser requests. An example is given below [66].

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

Listing 6: An external document defines an additional XML Entity send which is used for request forgery (see Listing 7)

If the Parameter Entity reads a local file, as illustrated in Listing 7, its content may then be sent to the attacker as a part of the URL's path or query-string.

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

Listing 7: The file hosted at attacker.org/ext.dtd concatenates the file content with a request URL using Parameter Entities.

The example above is a slight variation of a similar technique presented by Morgan and Ibrahim in 2014 [70].

4 Automated Analysis

The security evaluation of eID services shares many similarities with the analysis of SSO services, since both technologies are based on the same authentication protocols. This allows for already existing penetration testing tools to be used in the security evaluation. We decided to utilize Burp Suite (Burp),[1] which is a widely used penetration testing tool for web applications. Burp acts as an intercepting proxy and can be used to log, intercept, display, and modify HTTP traffic. To facilitate more complex scenarios, Burp offers extension points which allow developers to extend its existing functionality. One relevant extension we utilized was the **E**xtension for **P**rocessing and **Re**cognition of **S**ingle **S**ign-**O**n Protocols (EsPReSSO).[2] This extension is open-source and

[1] https://portswigger.net/

[2] https://github.com/RUB-NDS/BurpSSOExtension

is able to automatically identify and classify SSO messages, allowing penetration testers to easily analyze and manipulate SSO flows.

In this section, we first describe the basic functionalities of EsPReSSO which were available at the start of our project. Subsequently, the extensions we added to the tool to improve its usability for eID analyses are presented.

4.1 EsPReSSO – Basic Functionalities

EsPReSSO was designed and implemented to recognize and distinguish SSO protocols. It has an automatic scanning function that passively inspects the browser's traffic by scanning HTTP parameters and keywords. In the event that an SSO protocol is recognized, the request/response is highlighted and a note referencing to the protocol is shown. Furthermore, specifically for the SAML protocol, EsPReSSO provides a SAML-Editor and SAML-Attacker.

SAML-Editor. The SAML-Editor searches each intercepted HTTP request/response for SAML relevant parameters and automatically decodes the SAML message before displaying the SAML *AuthnReq* or *AuthnResponse*, respectively. Furthermore, the SAML-Editor provides the means to carry out modifications of the intercepted messages in a user-friendly way.

SAML-Attacker. EsPReSSO also provided a user interface with a small predefined set of attacks; for example, specific attacks targeting XML Signature processing. However, not all known attack vectors were covered by EsPReSSO.

4.2 Extending EsPReSSO

The shortcomings of the previous EsPReSSO version and the missing attacks were the motivating factors for the improvement of this extension.

SAML-Editor. SAML messages are transmitted in an encoded form. To alter an intercepted SAML message, a penetration tester needed to first decode the message, then manipulate it, and finally re-encode it before relaying the modified message. During security evaluations, this process can be time consuming and bothersome. We therefore addressed this issue by extending the SAML-Editor in EsPReSSO to achieve more fine-grained control over each aspect of the intercepted message. As an example, it is now possible to easily change the HTTP method (HTTP-GET or HTTP-POST) with the message being automatically re-encoded.

We additionally implemented a Certificate Viewer to make certificate properties and key material provided within the SAML messages easily accessible. The Certificate Viewer greatly eases a quick quality inspection of the included certificates.

DTD-Attacker. To facilitate penetration tests, we implemented a new feature in EsPReSSO – the DTD-Attacker. Based on the attacks summarized in Section 3, we created a set of 18 attack vectors and implemented these into the DTD-Attacker. During testing, the penetration tester can tweak the selected attack vector within the DTD-Attacker. All modifications are automatically applied to the original message. An abstract overview of the DTD-Attackers UI is presented in Appendix C.

The DTD-Attacker allows for easy, vector-independent pre-configurations that are automatically applied to the selected vectors. For example, the complexity of the DoS attack vectors can be optimized by specifying a number of recursive entities and entity references. This allows testers to precisely measure the impact of the DoS attack.

Furthermore, for those attack vectors which cause URL invocation, the DTD-Attacker allows for the configuration of arbitrary URLs, which may be needed in order to evaluate the success of the attack. It is also possible to apply a specific encoding to the vector. This may allow penetration testers to bypass simple filter mechanisms that only work when encountering the standard XML character set, i.e., UTF-8.

For a comprehensive vulnerability analysis, a large number of attack vectors must be tested. For this reason, the DTD-Attacker can be used in a fully automated mode. Before starting the evaluation, the penetration tester only needs to configure a single parameter: the *attacker-listener* URL where the HTTP requests are sent. The provided URL is then automatically inserted into all attack vectors and sent to the target. If the target is vulnerable, the *attacker-listener* will receive a corresponding request from the target. This automated approach allows penetration testers to quickly determine vulnerable targets and what specific attack vectors the target may be vulnerable to.

SAML-Attacker. We also extended the EsPReSSO functionality for targeted SAML attacks. EsPReSSO supported the execution of basic Certificate Faking (CF) [37] and XML Signature Wrapping (XSW) attacks [64]; however, the implementations of these attacks contained several bugs. We resolved these bugs and additionally implemented a new user interface for the Signature Exclusion (∅Sig) attack [37].

During execution of the ∅Sig or CF attack, the auditor can now select the specific signature element on which the attack will be applied. During the CF attack, the original certificate is copied, the key is replaced, and the certificate is re-signed. Next, the original certificate is replaced and the target message is also re-signed.

eIDAS Provider	Recursive Entities	External (Parameter) Entities	External (Parameter) Entities	SchemaLocation / XInclude	External DTD	File Exfiltration
	DoS		SSRF			File Exfiltration
eIDAS Pilot Sweden	–	✗[1]	✗	✓	✗	✗
eIDAS Pilot Belgium	–	✓	✓	✓	✓	✓
eIDAS Pilot Czech Republic	–	✓	✓	✓	✓	✓
eIDAS Pilot Denmark	–	✗[1]	✗	✓	✗	✗
eIDAS Pilot Estonia	–	✓	✓	✓	✓	✓
eIDAS Pilot France	–	✓	✓	✓	✓	✓
eIDAS Pilot Norway	–	✓	✓	✓	✓	✓
ArubaPEC S.p.A	–	✓	✓	✓	✓	✓
Intesa S.p.A.	–	✗[1]	✗[2]	✓	✗[2]	✓
InfoCert S.p.A.	–	✗[1]	✗	✓	✗	✗
Namirial	–	✗[1]	✗	✓	✗	✗
Poste Italiane SpA	–	✗[1]	✗[2]	✓	✗[2]	✓
Register.it S.p.A	–	✗[1]	✗	✓	✗	✗
Sielte S.p.A	–	✓	✓	✓	✓	✓
TI Trust Technologies srl (TIM)	–	✓	✓	✓	✓	✓
Vulnerable in Total	**–**	**7**	**7**	**0**	**7**	**5**

[1] To avoid harm, we did not test the full impact of the attacks.
[2] Only DNS requests were observed.

✓= Not vulnerable, ✗= Vulnerable, – = Not evaluated

Table 1: XML parsing vulnerabilities are still an effective attack technique to which 7 of the 15 tested SAML endpoints were found to be vulnerable.

5 Evaluation of XML Attacks

In our evaluation, we concentrated on the security analysis of general XML-based parsing attacks and their application to eIDAS services. This is because XML-based attacks do not necessarily demand the usage of valid accounts since only XML parsers are targeted. For their evaluation, only correct endpoint URLs of the analyzed service are necessary in order to gain access to the SAML parsing functionality. No valid eID cards, accounts, or configurations are necessary. This allowed us to test a large number of eIDAS services.

Table 1 shows the results of our evaluation for the 15 tested eIDAS services. It confirms that XML parsing attacks are still a prevalent attack technique. We cooperated with the affected providers who were able to successfully apply countermeasures to these attacks. Several providers which have not yet rolled out fixes or have not responded to our emails have been anonymized in Table 1. In case of unresponsive providers, we contacted the responsible CERT team regarding the security issues and are currently awaiting further information regarding the status of the fixes.

DoS Attacks. We were able to confirm the existence of DoS vulnerabilities in seven services. This can be concluded from the fact that DoS attacks based on External (Parameter) Entities are always possible if the XML parser can be tricked into loading external files. We did not evaluate whether DoS attacks with nested or recursive entities were applicable. When testing SAML endpoints, there is usually no direct server feedback and one cannot observe whether nested entities are resolved. Therefore, all test vectors with a real DoS potential were omitted to avoid damaging the tested service endpoints.

SSRF Attacks. Of the tested servers, seven were vulnerable to SSRF attacks. These could be executed using External (Parameter) Entities and by loading external DTDs. None of the servers resolved `SchemaLocation` or `XInclude` elements.

File Access. Of the fifteen eID services, five were vulnerable to attacks where the parsers were forced to read external or local files. Although two additional servers were vulnerable to SSRF attacks, we were only able to force these servers to issue DNS queries. Reading remote external files was not possible.

Lessons Learned. Although DTD vulnerabilities have been known since 2005 and multiple security studies exist, it is surprising that such a large number of vulnerabilities were found.

The user agent identifiers included in forged server-side requests indicate that all implementations vulnerable against SSRF and DoS are programmed in Java. As found in a comprehensive evaluation of XML parsers [66] provided in 2016, all Java XML parsers are vulnerable against XML External Entity Attacks (XXEAs) in their default configuration. Consequently, services remain vulnerable if the responsible developers and administrators do not explicitly disable these insecure features.

We employed the blackbox approach for testing and do not know whether the vulnerable services share the same XML stack. Nevertheless, we believe that Java's insecure defaults are the underlying reason for the abundance of similar vulnerability patterns.

6 Comprehensive Evaluation of the eIDAS Swedish Pilot

This section summarizes the security test suite that can guide both developers and security auditors alike during their work with SAML based SSO environments. A condensed overview is given below and in Table 3.

6.1 Testing Methodology

We define a number of tests relevant for the eID services, summarize known attacks, and categorize these into three different classes. Since many of the targeted eID services are potentially closed-source web applications, the chosen testing methodology is a black-box approach. This allows us to define a generic test suite irrespective of a programming language, framework, or similar restrictions that might otherwise be imposed by the test-target.

Limitations. During this survey we encountered the following problems:

- National eID schemes generally require a valid eID card to complete the entire authentication process. Due to the fact that no test eID card was available, we limited our targets to demo services which allowed authentication without an eID card. For this reason, the actual eID based End-User authentication is not part of our evaluation. Please note that communication between eIDAS compliant parties is not affected by this limitation.
- We focused on SAML because it constructs the compatibility layer of eIDAS. National eID schemes may of course be based on different technologies which are not considered in this work.
- There is no public list of existing eID services. Finding such services can, therefore, be a challenging task.
- To avoid harm on the evaluated systems, we only verified the possibility to carry out DoS attacks and we did not actually carry out these attacks.
- During the course of the project, some of the systems were updated with newer versions of the applications. Therefore, some previously discovered security issues were fixed before reporting these to the developers.

Selected Test Target. We were not able to perform comprehensive penetration tests for all the eIDAS providers listed in Table 1. The reason is that we did not possess valid identity cards or other credentials for these providers which are necessary to gain authentication tokens for further SAML evaluations. Therefore, for our penetration test, we chose the eIDAS Pilot of the Swedish E-identification Board.[3] It provides a demo SP as well as a fictional sending member state *Test Country XX* to simulate eIDAS authentication at a compliant IdP without requiring a valid eID card. This test environment, available from `http://eidasweb.se`, allowed us to simulate an end-to-end eIDAS authentication involving six SAML endpoints (see Appendix B), inspect all the messages exchanged through the user agent between participating services, and above all, to employ our test-suite in the field.

6.2 eIDAS Test-Suite

Our tests can be roughly divided into three categories: (1) Transport Layer Security, (2) Web Application Security, and (3) Message-Level Security. We provide a digest of the main components that compose the test suite in the following sections.

6.2.1 Transport Layer Security

TLS [14] is a complex protocol used to protect message integrity and confidentiality on the underlying transport protocol, and plays an important role in secure communications. Several critical vulnerabilities have been found in Transport Layer Security (TLS) libraries in the past, ranging from implementation faults to cryptographic weaknesses (cf. [61, 42, 3, 7, 8]). These problems can largely be mitigated by simply deploying the latest version of the used TLS libraries where the security updates have been applied. However, care must still be taken to securely configure the library's properties.

We evaluated the security of the TLS endpoints using `TLS-Attacker`,[4] which is currently the most advanced and freely available tool to discover security issues in TLS. We tested the deployed TLS protocol versions

[3] `https://www.elegnamnden.se`
[4] `https://github.com/RUB-NDS/TLS-Attacker`

and cipher suites, properties of the used certificates, and specific TLS attacks. In particular, this included: DROWN [3], POODLE [43], Heartbleed [57], Bleichenbacher's attack [6, 7], the invalid curve attack [28], and the padding oracle attack [71]. No host of the Swedish eIDAS test infrastructure was found to be vulnerable to any of these attacks, nor did we detect insecure configurations.

6.2.2 Message Level Security

SSO protocols usually involve multi-party communication. Therefore, it is not sufficient to simply secure the underlying transport; the message itself must be protected against tampering and, if necessary, provide appropriate guarantees for confidentiality of its contents. SAML makes use of XML Signatures [25] and XML Encryption [17] to fulfill these requirements.

Signature Exclusion (∅Sig). Each recipient of a message must only accept a message if a valid signature is provided. If the application accepts messages which do not include a signature, the message could be altered and the authentication is broken.

None of the six SAML services participating in the Swedish eIDAS pilot accepted a SAML message without a signature.

Certificate Faking (CF). The process of replacing the `<Signature>` element of an XML message with a self-generated signature and key is termed *Certificate Faking* [37]. Each recipient of a message, where the integrity is protected by an XML signature, must ensure to exclusively use trusted keys for signature verification. In particular, keys included in the message must not be considered as trustworthy without further verification. If a single SAML service is vulnerable to Signature Faking, the authentication scheme is broken because a malicious user is able to forge arbitrary messages and identities.

The Swedish eIDAS test environment was found to be resistant against Signature Faking; no involved entity accepted self-generated or faked signatures.

XML Signature Wrapping (XSW). The XSW attack against the XML Signature specification was first published in 2005 [40]. The main idea behind this attack is to change the structure of the XML element tree in such a way that the application's business logic processes a different element than the signature verification logic. This allows an adversary to submit arbitrary content to the vulnerable service. Several high-profile sites and SAML libraries have been found to be vulnerable to XSW in the past [64].

We tested a number of XSW techniques against the examined SAML services and were not able to successfully execute an XSW attack.

XSLT Attack (XSLTA). XML Signatures rely on certain preprocessing routines called *transformations*. These are used to derive a canonical form of the XML document before computing or verifying the associated signature. Because the transformations are applied before the document's signature can be verified, an adversary can alter the signature's `<transformation>` elements. The XSLT Attack (XSLTA) makes use of this fact and, in the worst case, can lead to arbitrary code execution. The XML processor should, therefore, disable support for the Extensible Stylesheet Language Transformation (XSLT) and terminate validation upon receiving invalid `<transformation>` elements.

We examined all services of the Swedish eIDAS pilot for XSLTAs and none was found to be vulnerable.

XML Encryption Attack (XEA). Several vulnerabilities in XML Encryption implementations have been found in the past, such as attacks against CBC mode in symmetric ciphers and attacks against asymmetric RSA-PKCS1.5 encryption [29]. Backwards compatibility attacks against secure algorithms must also be taken into consideration. Successful attacks against XML Encryption would undermine the confidentiality goals of eIDAS' encrypted SAML assertions [18].

We could not successfully execute or even test the underlying XML Encryption implementation used by the Swedish eIDAS pilot. This is because we were unable to find a signature bypass against any of the participating SAML services and, therefore, could not alter the ciphertext of encrypted assertions. One endpoint was found to encrypt SAML Assertions using unauthenticated CBC mode, potentially enabling attacks on encrypted XML ciphertexts [29]. However, the enveloped XML signature applied to the document's root element mitigated any XML Encryption Attacks.

Replay Attack. In an SSO context, replay attacks target the multiple redemption of an authentication token, regardless of any existing freshness and lifetime restrictions. SSO tokens contain at least one parameter guaranteeing freshness and one defining the expiration time. It is up to the SP to implement this verification correctly. A special case of Replay Attacks exists in SAML, where the *AuthnReq* can contain parameters restricting the lifetime and guaranteeing freshness. The IdP should verify all relevant parameters.

Five out of the six tested SAML endpoints accepted each token exactly once. Freshness of the token is ensured by the `IssueInstant` attribute and the unique token `ID`. The demo SP allowed multiple successful redemptions of the same *AuthnResponse*, as long as the token is submitted within the correct valid session and within the assertion's validity period. We were not able to circumvent the freshness or lifetime validation of any SAML

endpoint participating in the authentication flow.

Token Recipient Confusion (TRC). As a multitude of SPs may exist, an authentication token must clearly indicate its intended destination; a token should only be valid for a single SP to ensure that a malicious SP can not redeem captured user tokens at other benign SPs.

Our tests were restricted to a single SP and no signature bypasses were found; therefore, we could not evaluate the TRC attack.

6.2.3 Web Application Security

In SSO and eID authentication schemes, the SP, IdP, and the eIDAS nodes are web applications. End-User authentication involves loading multiple sites such as the SP, several redirect pages, and sites specific to the eIDAS Nodes, e.g., the consent page. Thus, attacks which target the web application, such as Clickjacking, XSS, and CSRF, must be considered for all sites loaded during the authentication.

Clickjacking. The goal of a *Clickjacking* or *User-Interface Redressing* attack is to trick the user into unknowingly execute actions of the attacker's choice. To accomplish this, UA features like transparent `iFrames` and `Drag'n'Drop` mechanisms are employed, and are often combined with some pretended incentive for the end-user to induce the intended action [67, 68]. Successful Clickjacking attacks can bypass Same-Origin-Policy checks and circumvent CSRF protection mechanisms, resulting in data exposure or an account to become compromised. In SSO scenarios, Clickjacking vulnerabilities can be used to lure the victim into unknowingly authenticating at the IdP and authorizing the access to restricted/sensitive resources.

On this note, an early version of the Swedish eIDAS pilot was vulnerable to Clickjacking attacks due to missing HTTP Security-Headers. Any stage of the authentication process could be *framed* in a transparent `iFrame`. This vulnerability was mitigated before we could reach out to the developers by adding the appropriate `X-Frame-Options` header to relevant pages.

XSS. As one of the most common attacks in the modern web, according to [75], XSS has been studied extensively and yet, XSS vulnerabilities and circumventions of XSS protection mechanisms are still frequently found in complex web applications [41, 5, 23, 24, 33, 59]. A XSS vulnerability enables a malicious actor to inject arbitrary script code into a website's DOM. The script is subsequently executed by the victim's UA. Attacker controlled code running in the origin of the attacked page is a critical vulnerability which can bypass the security of the authentication scheme.

We carefully inspected the eIDAS test deployment of the Swedish E-Legitimation Board and did not find a XSS vulnerability on any of the sites involved in the simulated eIDAS authentication.

CSRF. In a Cross-Site Request Forgery (CSRF) attack a victim is tricked into unknowingly performing state-changing actions on a vulnerable site. To this end, an adversary injects malicious requests into the victim's UA which has an authenticated session with the vulnerable application. A CSRF attack abuses the fact that the UA automatically includes session credentials, more particularly cookies, in each request [75, 4]. In the past, CSRF was found to be exploitable in widely deployed SSO solutions [34, 35]. eIDAS mandates manual user authentication for each authentication request [20, Section 2.4.1]. This constraint must be implemented by every IdP. If this is not implemented, a victim can be unknowingly logged in at arbitrary services or be tricked into authorizing access to restricted resources.

We did not find any exploitable CSRF vulnerabilities in the Swedish eIDAS pilot. All critical components appeared to follow best current practices and *AuthnResponse*s were bound to session-specific random tokens.

Covert Redirect (CR). Some web applications store the URL navigated by the End-User before starting the SSO authentication and include this parameter as part of the *AuthnReq*, for example, as a GET parameter `next_url` or `RelayState`. After receiving the authentication token, the SP then forwards the user to the resources initially requested by the End-User. Unfortunately, during this forwarding, sensitive information can be leaked. As an example, in some SSO protocols, the `Referer` header can contain the authentication token. This can potentially lead to information leakage and broken authentication.

The Swedish eIDAS pilot and, in particular, the demo SP do not make use of parameter based redirects, and are not vulnerable to Covert Redirect attacks.

HTTP Security Headers. A number of security related HTTP headers have been defined. These mechanisms allow communicating entities to share security related information and influence security related behavior or decisions of each other. In this manner, HTTP security headers play an integral role in safeguarding today's web application security.

In our tests, we ascertained the presence and sound configuration of the HTTP headers listed below.

```
X-Frame-Options
Content-Security-Policy
Strict-Transport-Security
Content-Type
X-Content-Type-Options
X-Xss-Protection
Public-Key-Pins
```

Furthermore, the directives of the `Set-Cookie` header were checked. In particular, the `httpOnly` and the `secure` flags must be set if a cookie contains sensitive information such as session IDs. To prevent CSRF attacks, the `SAMESITE` directive may be added.

In Appendix A, we summarize best current practices regarding the security headers.

7 Related Work

In 2003 and 2006 Gross et al. [21, 22], and in 2008 and 2011 Armando et al. [2, 1], a formal model for the SAML Browser/Artifact profile was analyzed which identified several generic flaws allowing for connection hijacking/replay, Man-in-the-Middle (MitM), and HTTP referrer attacks. In 2012, Somorovsky et al. [64, 63] investigated the XML Signature validation of several SAML frameworks and web services, discovering critical flaws based on XSW. In 2014, Mainka et al. [37] analyzed 22 Cloud 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. Mayer et al. [39] discovered, in 2014, critical vulnerabilities in SAML IdPs by exploiting XSS vulnerabilities and flaws in the SAML implementation. In 2016, Späth et. al [66] provided a comprehensive security analysis of XML parsers regarding their security against XML-based attacks such as XML External Entity. This survey provides a comprehensive summary of attack vectors which we used during our evaluation. In 2016 and 2017, Kakavas et al. and Sanso et al. discovered critical vulnerabilities in prominent web applications like Office365 [30], GitHub [31], and Slack [58] by using already known attack vectors. In 2018, two novel attack vectors were discovered by RedTeam [56] and Duo [36]. Both vectors used a truncation technique to insert malicious identities within the authentication tokens without invalidating the digital signature. However, none of the previous security researches covered eID services and evaluated the security of the used authentication protocols and the web interfaces.

The document published by the European Commission on *eIDAS-Node Security Considerations* [13] describes the security best practices for eIDAS infrastructures. However, this document mostly concentrates on best practices for typical web attacks, and summarizes secure usage of HTTP headers and key storage. In our paper, we also provide an overview of SAML and XML-relevant attacks, and summarize best practices for these technologies. Our study is based on many relevant recommendations issued by OWASP [50, 49, 53, 45, 46] and the BSI [11, 10].

8 Conclusion

To the best of our knowledge, we provided the first security analysis of the eIDAS infrastructure and trust services. We find it impressive that many known attacks from previous works were not applicable to the examined eIDAS services. We consider this to be a positive result of applying proper countermeasures and following the current security best practices.

Nevertheless, we were able to exploit XML parsing vulnerabilities on about half of the tested services. The fact that most of the vulnerable services appear to be implemented in Java highlights the importance of secure defaults as well as the consequences this poses to production systems.

Our survey reveals the complexity of current authentication systems, which is a natural consequence of the complex technology stack in use. Peculiarities of TLS, XML, SAML, and HTML/JavaScript/AJAX must be considered, and each of these technologies must be strengthened against potential attacks. Additionally, interactions of the various layers and potential security relevant consequences must be taken into account. In our paper, we show again how the insecurity of one component can bypass the security of the entire system, even if all other components are secure.

Furthermore, our study made clear that a demand still exists for tools which facilitate automatic security analyses. Similarly, carefully compiled documents specifying security best practices appear to be lacking. We hope that our tool and the Best Current Practices document can be used as a foundation for future security researchers to fill this documentational gap.

Aside from the technical issues we resolved during this evaluation, one major obstacle was in obtaining valid credentials for testing purposes. We could apply the full test suite to only the Swedish eIDAS Pilot because it provided a simulation of the eID based End-User authentication. In order to execute a more widespread evaluation, obtaining valid credentials may be necessary.

We recognize and appreciate that an increasing number of eIDAS related projects are publishing their work as open-source. This openness enables future researchers to use techniques like white-box testing and static code analysis to complement our black-box approach and further increase the security of eID services.

Acknowledgements

The research was 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 helpful discussions provided.

References

[1] ARMANDO, A., CARBONE, R., COMPAGNA, L., CUELLAR, J., PELLEGRINO, G., AND SORNIOTTI, A. From multiple credentials to browser-based single sign-on: Are we more secure? In *IFIP International Information Security Conference* (2011), Springer Berlin Heidelberg, pp. 68–79.

[2] ARMANDO, A., CARBONE, R., COMPAGNA, L., CUELLAR, J., AND TOBARRA, L. Formal analysis of saml 2.0 web browser single sign-on: Breaking the saml-based single sign-on for google apps. In *Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering* (New York, NY, USA, 2008), FMSE '08, ACM, pp. 1–10.

[3] AVIRAM, N., SCHINZEL, S., SOMOROVSKY, J., HENINGER, N., DANKEL, M., STEUBE, J., VALENTA, L., ADRIAN, D., HALDERMAN, J. A., DUKHOVNI, V., KÄSPER, E., COHNEY, S., ENGELS, S., PAAR, C., AND SHAVITT, Y. DROWN: Breaking TLS Using SSLv2. In *25th USENIX Security Symposium (USENIX Security 16)* (Austin, TX, Aug. 2016), pp. 689–706.

[4] BARTH, A., JACKSON, C., AND MITCHELL, J. C. Robust defenses for cross-site request forgery. In *Proceedings of the 15th ACM conference on Computer and communications security* (2008), ACM, pp. 75–88.

[5] BATES, D., BARTH, A., AND JACKSON, C. Regular expressions considered harmful in client-side XSS filters. In *Proceedings of the 19th international conference on World wide web* (New York, NY, USA, 2010), WWW '10, ACM, pp. 91–100.

[6] BLEICHENBACHER, D. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *Advances in Cryptology – CRYPTO '98*, vol. 1462 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 1998.

[7] BÖCK, H., SOMOROVSKY, J., AND YOUNG, C. Return of bleichenbacher?s oracle threat (robot).

[8] BÖCK, H., ZAUNER, A., DEVLIN, S., SOMOROVSKY, J., AND JOVANOVIC, P. Nonce-disrespecting adversaries: Practical forgery attacks on gcm in tls. *IACR Cryptology ePrint Archive 2016* (2016), 475.

[9] BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, C. M., MALER, E., AND YERGEAU, F. Extensible Markup Language (XML) 1.0 (Fifth Edition). *W3C Recommendation* (2008).

[10] BSI. Technical guideline tr-03130 eid-server. part 1: Functional specification, Oct. 2017.

[11] BSI. Technische richtlinie tr-02102-1: Kryptographische verfahren: Empfehlungen und schlüssellängen, Jan. 2018.

[12] CANTOR, S., KEMP, J., PHILPOTT, R., AND MALER, E. Assertions and protocols for the oasis security assertion markup language (saml) v2.0, Mar. 2005.

[13] COMMISSION, E. eidas-node security considerations, version 1.0, 2018.

[14] DIERKS, T., AND RESCORLA, E. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008. Updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685, 7905, 7919.

[15] EASTLAKE, D., REAGLE, J., HIRSCH, F., ROESSLER, T., IMAMURA, T., DILLAWAY, B., SIMON, E., YIU, K., AND NYSTRÖM, M. XML Encryption Syntax and Processing 1.1. *W3C Candidate Recommendation* (2012). `http://www.w3.org/TR/2012/WD-xmlenc-core1-20121018`.

[16] EASTLAKE, D., REAGLE, J., SOLO, D., HIRSCH, F., AND ROESSLER, T. XML Signature Syntax and Processing (Second Edition). W3C Recommendation, June 2008. `http://www.w3.org/TR/xmldsig-core/`.

[17] EASTLAKE, D., REAGLE, J., SOLO, D., HIRSCH, F., AND ROESSLER, T. XML Signature Syntax and Processing (Second Edition). *W3C Recommendation* (2008).

[18] (EU), C. C. E. eidas - cryptographic requirements for the interoperability framework - tls and saml. `https://ec.europa.eu/cefdigital/wiki/download/attachments/46992719/eidas_-_crypto_requirements_for_the_eidas_interoperability_framework_v1.0.pdf`, 2015. Last accessed: 24.5.2018.

[19] (EU), C. C. E. eidas - interoperability architecture version 1.00. `https://ec.europa.eu/cefdigital/wiki/download/attachments/46992719/eidas_interoperability_architecture_v1.00.pdf`, 2015. Last accessed: 24.5.2018.

[20] (EU), C. C. E. F. eidas saml message format - version 1.0. `https://ec.europa.eu/cefdigital/wiki/download/attachments/46992719/eidas_message_format_v1.0.pdf`, 2015. Last accessed: 24.5.2018.

[21] GROSS, T. Security analysis of the saml single sign-on browser/artifact profile. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual* (2003), IEEE, pp. 298–307.

[22] GROSS, T., AND PFITZMANN, B. Saml artifact information flow revisited. In *In IEEE Workshop on Web Services Security (WSSS)* (2006), pp. 84–100.

[23] GUPTA, S., AND GUPTA, B. B. Cross-site scripting (xss) attacks and defense mechanisms: classification and state-of-the-art. *International Journal of System Assurance Engineering and Management 8*, 1 (Jan 2017), 512–530.

[24] HEIDERICH, M., SCHWENK, J., FROSCH, T., MAGAZINIUS, J., AND YANG, E. Z. mxss attacks: Attacking well-secured web-applications by using innerhtml mutations. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security* (New York, NY, USA, 2013), CCS '13, ACM, pp. 777–788.

[25] HIRSCH, F., SOLO, D., REAGLE, J., EASTLAKE, D., AND ROESSLER, T. XML Signature Syntax and Processing (Second Edition). W3C recommendation, W3C, June 2008.

[26] JAGER, T., PATERSON, K. G., AND SOMOROVSKY, J. One Bad Apple: Backwards Compatibility Attacks on State-of-the-Art Cryptography. In *Network and Distributed System Security Symposium (NDSS)* (February 2013).

[27] JAGER, T., SCHINZEL, S., AND SOMOROVSKY, J. Bleichenbacher's attack strikes again: Breaking pkcs#1 v1.5 in xml encryption. In *ESORICS* (2012), pp. 752–769.

[28] JAGER, T., SCHWENK, J., AND SOMOROVSKY, J. Practical Invalid Curve Attacks on TLS-ECDH. *20th European Symposium on Research in Computer Security (ESORICS)* (2015).

[29] JAGER, T., AND SOMOROVSKY, J. How To Break XML Encryption. In *The 18th ACM Conference on Computer and Communications Security (CCS)* (Oct. 2011).

[30] KAKAVAS, I. The road to hell is paved with saml assertions, 2016.

[31] KAKAVAS, I. The road to your codebase is paved with forged assertions, 2017.

[32] KLEIN, A. Klein: Multiple vendors xml parser (and soap/web- services server) denial of service attack using dtd., 2002.

[33] LEKIES, S., KOTOWICZ, K., GROSS, S., VELA NAVA, E. A., AND JOHNS, M. Code-reuse attacks for the web: Breaking cross-site scripting mitigations via script gadgets. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), ACM, pp. 1709–1723.

[34] LI, W., AND MITCHELL, C. J. Security issues in oauth 2.0 sso implementations. In *International Conference on Information Security* (2014), Springer, pp. 529–541.

[35] LI, W., AND MITCHELL, C. J. Analysing the security of google's implementation of openid connect. In *Proceedings of the 13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment - Volume 9721* (New York, NY, USA, 2016), DIMVA 2016, Springer-Verlag New York, Inc., pp. 357–376.

[36] LUDWIG, K. Duo finds saml vulnerabilities affecting multiple implementations, February 2018.

[37] MAINKA, C., MLADENOV, V., FELDMANN, F., KRAUTWALD, J., AND 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* (2014), CCSW '14.

[38] MARSH, J., ORCHARD, D., AND VEILLARD, D. Xml inclusions (xinclude). *W3C, version 1* (2006).

[39] MAYER, A., NIEMIETZ, M., MLADENOV, V., AND SCHWENK, J. Guardians of the clouds: When identity providers fail. In *Proceedings of the 6th Edition of the ACM Workshop on Cloud Computing Security* (New York, NY, USA, 2014), CCSW '14, ACM, pp. 105–116.

[40] MCINTOSH, M., AND AUSTEL, P. XML Signature Element Wrapping Attacks and Countermeasures. In *SWS '05: Proceedings of the 2005 workshop on Secure web services* (New York, NY, USA, 2005), ACM Press, pp. 20–27.

[41] MELICHER, W., DAS, A., SHARIF, M., BAUER, L., AND JIA, L. Riding out domsday: Towards detecting and preventing dom cross-site scripting. *Network and Distributed Systems Security (NDSS) Symposium 2018* (2018).

[42] MEYER, C. *20 Years of SSL/TLS Research : An Analysis of the Internet's Security Foundation*. PhD thesis, Ruhr-University Bochum, Feb. 2014.

[43] MÖLLER, B., DUONG, T., AND KOTOWICZ, K. This poodle bites: exploiting the ssl 3.0 fallback. *Security Advisory* (2014).

[44] MOZILLA. Content-security-policy - http | mdn, 2018.

[45] OWASP. Content security policy cheat sheet, 2015.

[46] OWASP. Saml security cheat sheet, 2017.

[47] OWASP. Clickjacking defense cheat sheet, April 2018.

[48] OWASP. Content security policy scanner, April 2018.

[49] OWASP. Owasp secure headers project, April 2018.

[50] OWASP. Samesite, 2018.

[51] OWASP. Session management cheat sheet, April 2018.

[52] OWASP. Tls cheat sheet, April 2018.

[53] OWASP. Xml external entity (xxe) prevention cheat sheet, 2018.

[54] PARLIAMENT, E., AND UNION, T. C. O. T. E. Regulation (eu) no 910/2014 of the european parliament and of the council. `http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32014R0910&from=EN`, 2014.

[55] PORTSWIGGER. Burpsuite, April 2018.

[56] REDTEAM. Truncation of saml attributes in shibboleth 2, January 2018.

[57] RIKU, ANTTI, MATTI, AND MEHTA. Heartbleed, cve-2014-0160, 2015. `http://heartbleed.com/`.

[58] SANSO, A. Slack saml authentication bypass, October 2017.

[59] SCALZI, G. Content-security-policy: Misconfiguration and bypasses, 2016.

[60] SCOTT CANTOR, FREDERICK HIRSCH, J. K. R. P. E. M. Bindings for the oasis security assertion markup language (saml) v2.0, March 2005.

[61] SHEFFER, Y., HOLZ, R., AND SAINT-ANDRE, P. Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS). RFC 7457 (Informational), Feb. 2015.

[62] SOMOROVSKY, J. On the Insecurity of XML Security (Doctoral dissertation). Ruhr University Bochum, Germany, July 2013.

[63] SOMOROVSKY, J., HEIDERICH, M., JENSEN, M., SCHWENK, J., GRUSCHKA, N., AND 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).

[64] SOMOROVSKY, J., MAYER, A., SCHWENK, J., KAMPMANN, M., AND JENSEN, M. On breaking saml: Be whoever you want to be. In *In Proceedings of the 21. USENIX Security Symposium* (Bellevue, WA, Aug. 2012).

[65] SPÄTH, C. Security implications of dtd attacks against a wide range of XML parsers. Master, Ruhr-University Bochum, Oktober 2015.

[66] SPÄTH, C., MAINKA, C., MLADENOV, V., AND SCHWENK, J. Sok: Xml parser vulnerabilities. In *10th USENIX Workshop on Offensive Technologies (WOOT 16), Austin, TX* (2016).

[67] STONE, P. Next generation clickjacking, April 2010.

[68] STUTTARD, D., AND PINTO, M. *The web application hacker's handbook: Finding and exploiting security flaws*. John Wiley & Sons, 2011.

[69] SULLIVAN, B. Security briefs - xml denial of service attacks and defenses. `https://msdn.microsoft.com/en-us/magazine/ee335713.aspx`, November 2009. Last accessed: 20.5.2018.

[70] TIMOTHY D. MORGAN, O. A. I. Xml schema, dtd, and entity attacks. Tech. rep., VSR, May 2014.

[71] VAUDENAY, S. Security Flaws Induced by CBC Padding – Applications to SSL, IPSEC, WTLS... In *Advances in Cryptology – EUROCRYPT 2002*, vol. 2332 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, Apr. 2002.

[72] W3AF. Web application attack and audit framework (w3af), April 2018.

[73] WEICHSELBAUM, L. Csp evaluator, 2016.

[74] WEICHSELBAUM, L., SPAGNUOLO, M., LEKIES, S., AND JANC, A. Csp is dead, long live csp! on the insecurity of whitelists and the future of content security policy. In *Proceedings of the 23rd ACM Conference on Computer and Communications Security* (Vienna, Austria, 2016).

[75] WICHERS, D. Owasp top ten project. Tech. rep., OWASP, Sept. 2015.

[76] ZALEWSKI, M. *The tangled Web: A guide to securing modern web applications*. No Starch Press, 2012.

Appendices

A Best Current Practices

In this section, we give an overview of the best current practices which should be considered during the implementation of SSO services. We provide existing BCP documents on this topic and an overview on existing penetration testing tools.

A.1 Transport Layer Security

When enabling TLS, the following security checks need to be considered:

- Only secure TLS versions must be used – TLS 1.2 and 1.3. TLS 1.0 and 1.1 may be used; however, they are not recommended.
- Only secure cipher suites must be activated. Detailed recommendations are provided by OWASP [52].
- Disabling TLS compression: Activating TLS compression could make the running service implementation vulnerable to the CRIME attack [52].

There are different online services and tools for evaluating the security of TLS configurations, such as: SSL Labs,[5] testssl.sh,[6] or a TLS scanner based on TLS-Attacker.[7] We recommend their usage after successful TLS deployment.

Cryptographic Key Lengths and Algorithms. The following cryptographic algorithms and key lengths are relevant for the deployment of SAML and TLS:

- Key lengths: RSA – 2048 bit; DH/DSS – 2048 bit; ECDH/ECDSA – 256 bit
- Eliptic curves: secp256r1, secp384r1, secp521r1, brainpoolP256r1, brainpoolP384r1, brainpoolP512r1
- Hash algorithms: SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512

More details regarding the restrictions are published in [11].

[5] https://www.ssllabs.com/ssltest/
[6] https://testssl.sh/
[7] https://github.com/RUB-NDS/TLS-Scanner

X.509 Certificates. X.509 certificates are used in TLS as well as in SAML. The best practices for processing and issuing X.509 certificates can be summarized as follows:

- Trust establishment: X.509 certificates must be issued by trusted authorities located in the truststore. Trust validation must be enforced.
- Updating certificates: A process of updating certificates before their expiration must be established.
- Avoiding wildcard certificates: Certificates with wildcards in the subject, common name or alternative names should be avoided.

A.2 Message Level Security

XML Parser. We strongly recommend disabling the following features within the parser:

- DTD processing. This feature should be only activated if it is needed.
- Disabling the network access to the parser. Aside from processing DTDs, there are further possibilities to call arbitrary URLs. By disabling network access, calling these URLs can no longer occur.
- If DTDs cannot be disabled, imposing restrictions on processing entities must be done by: (1.) Limiting the memory capacity that a parser can allocate (2.) Disabling the SYSTEM and PUBLIC usage for all types of entities (internal and external parameter/general entities).

A comprehensive description of countermeasures and parser configurations is discussed by Späth et al. [65, 66].

XML Signatures. The best practices for processing XML Signatures in SAML messages can be depicted as follows:

- Signature Exclusion Attacks: It must be ensured that the data is signed and that the signature has not been removed.
- XML Signature Wrapping (XSW) Attack: During verification, it must be verified that the signature has been constructed over the processed data. More concrete countermeasures are discussed by Somorovsky et al. [62, 64, 46]. The XML Signature specification also provides additional recommended security considerations [17].
- Certificate Validation: The certificate used for signature generation must be issued by a trusted IdP.
- XSLT: It is not allowed to trigger the XSLT processor during any XML Signature transformation.

XML Encryption. The newest XML Encryption stan-

dard provides best practices for a secure standard deployment. These can be summarized as follows [15]:

- A SAML server implementing XML Encryption and XML Signature should use at least two distinct certificates. It is good cryptographic practice to use different keys for different purposes; in this case, for decryption of encrypted XML contents and for signing SAML messages. If not implemented, backwards compatibility attacks could be executed [26].

- To protect against adaptive chosen-ciphertext attacks on symmetric encryption schemes [29], authenticated encryption schemes should be used. XML Encryption 1.1 provides the AES-GCM algorithms. Other algorithms should not be supported. If they are supported, it must be ensured the attacker cannot enforce processing of unauthenticated XML ciphertexts by the server [62].

- To protect against adaptive chosen-ciphertext attacks on asymmetric encryption schemes [27], secure encryption schemes must be used: RSA-OAEP and elliptic curve Diffie-Hellman. Other algorithms should not be supported. If they are supported, specific countermeasures must be applied, most importantly, against Bleichenbacher's attack [62].

Further security best practices are located in the XML Encryption specification [15].

SAML Validation. The following aspects must be considered by validating the `SAMLRequest`:

- `AssertionConsumerServiceURL`: The URL must be checked against a whitelist with pre-defined URLs. Usually, this whitelist is provided by the metadata of the provider.

- Freshness validation: All timestamps located in the message must be valid.

The following security aspects are relevant to the `SAMLResponse`:

- Issuer validation: The SAML issuer (IdP) must be validated against a whitelist of trusted IdPs allowed to authenticate the users.

- Recipient validation: The SAML recipient must be validated by comparing the value with the expected recipient of this message. In case of deviations, the message must be rejected.

- Freshness validation: To prevent replay attacks, the signed timestamps must be validated.

- `InResponseTo` validation: It must be checked whether the content of the `InResponseTo` element is identical to the content of the `ID` sent in the *AuthnReq*. Otherwise, CSRF attacks can be applied.

Further security considerations can be found in [46].

A.3 Web Application Security

In this section, we provide a summary of the security relevant HTTP headers which should be configured to strengthen the communication between the provider and the End-User's UA. A comprehensive summary is provided by OWASP in [49].

HTTP Session Cookies. The security of session cookies is essential for the correct End-User authentication. In the event of misconfiguration, an attacker could hijack the authenticated HTTP session of an End-User and impersonate them. We consider the following cookie flags as required: `secure` and `HttpOnly` [51]. In addition, `samesite` cookies can be applied to reduce the risk against CSRF attacks.

A common pitfall lies within the header's `domain` directive, which broadens the cookie's scope to include the originating host's sub-domains and may lead to unintended data exposure. The other cookie-scoping directive, `path`, should not be used for security relevant scoping [76].

Clickjacking/UI-Redressing. The main goal of the proposed countermeasures is to prevent framing a website within another one. By this means, attacks such as Clickjacking and UI-redressing can be mitigated. They can be prevented by using one of the following techniques: `X-Frame-Options` HTTP header, the Content-Security-Policy, or JavaScript code [47].

HTTP Strict Transport Security. Securing the communication between the UA and the server is essential with respect to eavesdropping attacks. For this purpose, the use of TLS is imperative. By using the `Strict-Transport-Security` header the server can force the UA to always use TLS. In this way, the risk against man-in-middle attacks can be reduced.

Content Security Policy (CSP). The Content Security Policy is a powerful construct. The specified directives and configuration possibilities provide the means to mitigate XSS vulnerabilities, protect against Clickjacking, Mixed-Content inclusion, and generally restrict client-side resource inclusion [45, 44, 74]. However, the CSP is a *defense-in-depth* approach that requires additional effort from web-developers. As an example, neither in-line scripts nor event handlers can be used without additional measures.

Specific configuration of a web application's CSP depends on multiple factors. These factors include: the current version of the CSP, design and architecture of the website, required external resources from different domains, and the general complexity of the web application. Therefore, it is not possible to give a general-

SAML Endpoints	
URL	**Tag**
`eunode.qa.sveidas.se/idp/profile/SAML2/POST/SSO`	H_1
`nonode.eidastest.se/EidasNode/ColleagueRequest`	H_2
`nonode.eidastest.se/PS-IdP/AuthenticateCitizen`	H_3
`nonode.eidastest.se/EidasNode/IdpResponse`	H_4
`eunode.qa.sveidas.se/idp/extauth/saml2/post`	H_5
`eunode.eidastest.se/con-sp/assertionconsumer`	H_6

Table 2: Mapping of SAML endpoint URLs to tags used in Table 3.

purpose recommendation of a *good policy*.

Tool Support. Several software products can support developers in evaluating their applications. The ZAP Content Security Policy Scanner extension is able to provide an automated analysis of the security headers, evaluate the applied Content Security Policy [48], and find potential XSS, CSRF, and Clickjacking attacks. Similar extensions exist for Burp-Suite and w3af [55] [72]. The authors of [74] provide an online tool for CSP evaluation [73].

B Swedish eIDAS Pilot - Message Flow

The authentication scenario depicted in Figure 3 assumes that a Swedish SP `eunode.eidastes.se/con-sp` requests authentication from a user in *Test Country XX* by reaching out to the eIDAS Connector (`eunode.qa.sveidas.se`). The *AuthnReq* is relayed to the eIDAS Proxy-Service at `nonode.eidastest.se/EidasNode` and eventually forwarded to Test Country XX's IdP. After successful user authentication, as simulated in the test-pilot, the *Authn-Response* is delivered backwards through the same channel. Explicit user consent is required before the *Authn-Response* is released by the Proxy-Service.

C DTD-Attacker in EsPReSSO

Figure 4 presents the manual interface to the novel DTD-Attacker. The user can choose from a variety of different attack vectors and easily modify the provided templates as required.

D Security Evaluation - Summary

A summary of our results is given in Table 3 and Table 2 for the security analysis performed on the Swedish eIDAS demo service.

	Transport Layer Security									Message Level Security								Web Application Security				
SAML Endpoint	***TLS Version***	***Cipher Suites***	***Certificate***	***DROWN***	***POODLE***	***Bleichenbacher***	***Padding Oracles***	***Invalid Curves***	***Heartbleed***	***∅Sig***	***CF***	***XSW***	***XEA***	***Replay Attack***	***TRC***	***XSLT***	***External Entity Attack***	***XSS***	***CSRF***	***Clickjacking***	***Covert Redirect***	***Headers***
H_1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
H_2*	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗[1]	✓	✓[2]
H_3*	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗[1]	✓	✓[2]
H_4*	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗[1]	✓	✓[2]
H_5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓[4]	✓	✓	✓	✓	✓	✓	✓	✓	✓
H_6*	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓[3]	✓	✓	✓

* Shared TLS Endpoint of `eunode.eidastest.se` and `nonode.eidastest.se` (virtual hosting with shared certificate)
[1] Attack mitigated in newer versions
[2] Missing `X-Frame-Options` header and `frame-ancestor` directive in `CSP`
[3] `RelayState` parameter not properly bound to session
[4] Encrypts Assertion using AES128-CBC (no ciphertext authentication)

✓= Not vulnerable, ✗= Vulnerable, ✓= Weak configuration, ✗= Vulnerability mitigated
H1-H6: See Table 2 for specific SAML endpoint URLs.

Table 3: Results of the security evaluation of the Swedish eIDAS pilot. User authentication was simulated using the provided IdP of fictitious *Test Country XX*.

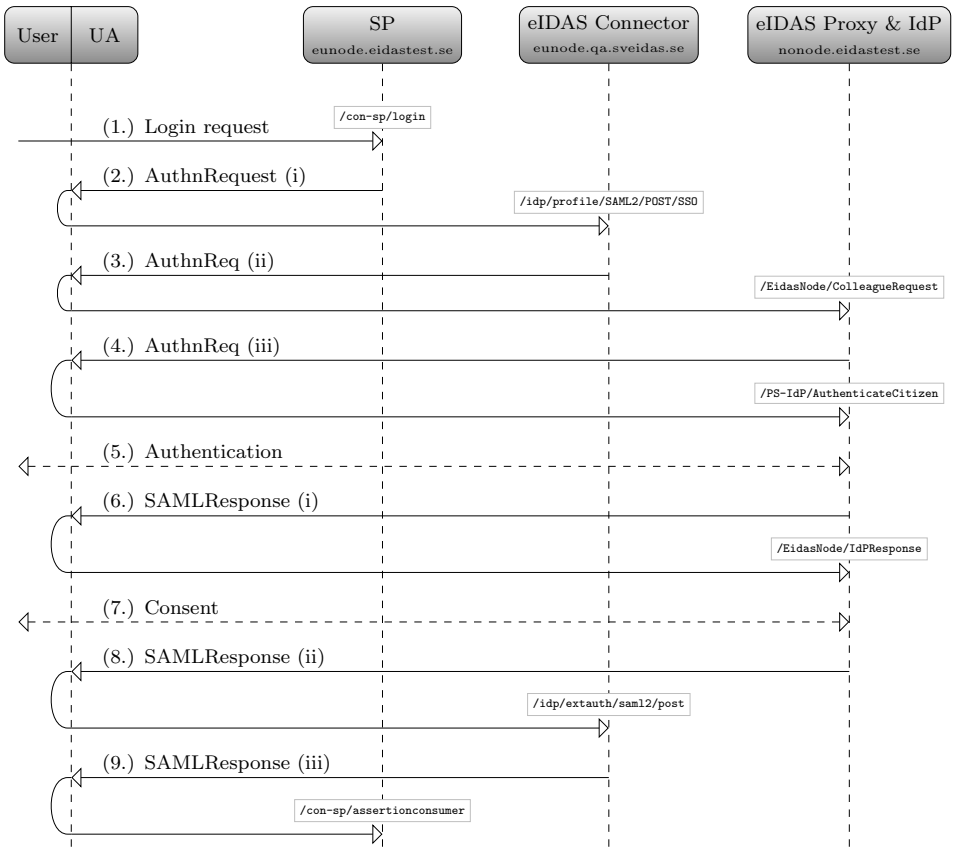

Figure 3: Message flow of an authentication process at the Swedish eIDAS pilot. The eIDAS connector represents the receiving member-state (Sweden) while the Proxy-Service and IdP represent the sending member-state (Test Country XX).

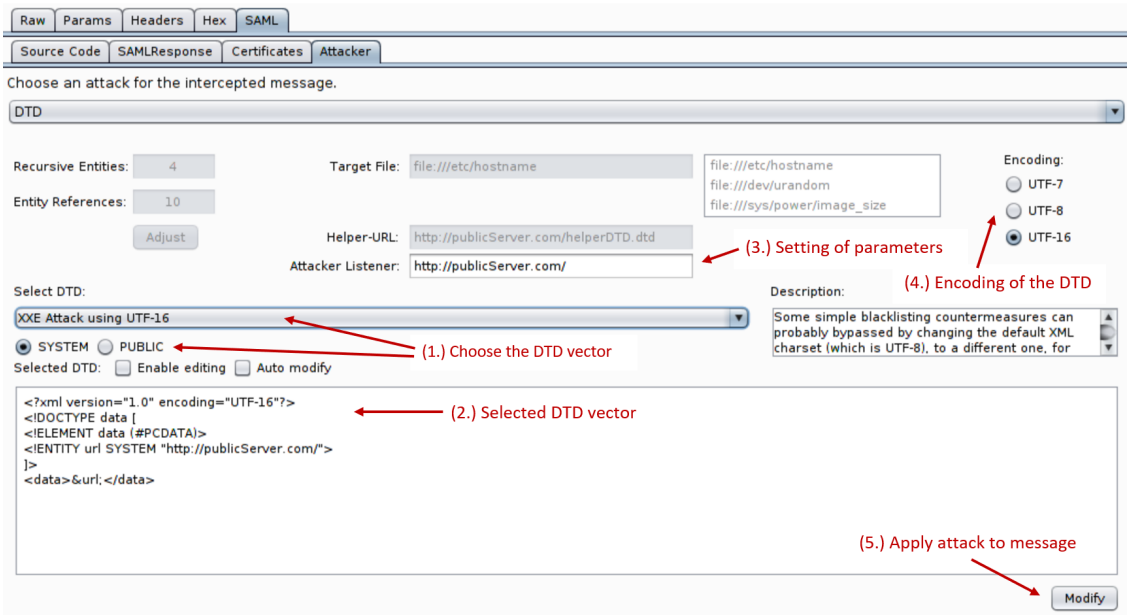

Figure 4: DTD-Attacker is a novel enhancement of the Burp plugin EsPReSSO. The manual mode provides predefined attack vectors which can easily be configured in every detail.