Security Analysis of the SAML Single Sign-on Browser/Artifact Profile

Thomas Groß
IBM Zurich Research Laboratory
tgr@zurich.ibm.com

Abstract

Many influential industrial players are currently pursuing the development of new protocols for federated identity management. The Security Assertion Markup Language (SAML) is an important standardized example of this new protocol class and will be widely used in business-to-business scenarios to reduce user-management costs. SAML utilizes a constraint-based specification that is a popular design technique of this protocol class. It does not include a general security analysis, but provides an attack-by-attack list of countermeasures as security consideration. We present a security analysis of the SAML Single Sign-on Browser/Artifact profile, which is the first one for such a protocol standard. Our analysis of the protocol design reveals several flaws in the specification that can lead to vulnerable implementations. To demonstrate their impact, we exploit some of these flaws to mount attacks on the protocol.

1. Introduction

One of the most important problems in the network-oriented industry currently is the reduction of user-management costs. Thus, many influential industrial players strive for the development of new protocols for federated identity management. Using these protocols, the companies are able to simplify user-management in an increasingly dynamic world and to benefit from user registrations done by other companies. The newly developed protocols will widely be used in business-to-business scenarios to allow the federation of inter-company services and to provide access control for supply-chain partners. Thus, major players in the access control market currently include these protocols in their products.

One of the most important proposals in this area is the Security Assertion Markup Language (SAML) [7, 8]. SAML is a very extensible, open standard, which makes it attractive as a basis for further development. Thus, various protocols of the Liberty Alliance Project [11, 20] and the Shibboleth Project [4] build on SAML.

The salient feature of most of these protocols is that they only require a standard Web browser as a user agent. We call this protocol class *browser-based* or *zero-footprint*. This feature is motivated by the fact that most potential users do not want to install protocol-specific software. Furthermore, it is desirable that the protocols do not require active content or cookies, because many users are not willing to use them for security or privacy reasons. Given these restrictions, the protocol designers have to work with browser redirects and HTTP constructs only, which implies new requirements that have not been considered by prior research.

In this paper, we analyze the SAML Single Sign-on Browser/Artifact profile, a three-party authentication protocol. Such a single sign-on protocol allows a user to sign-on only at his or her identity supplier, which in turn confirms the user's identity to other parties. As the protocol is part of the only open standard in this area and does not rely on active content or cookies, it is one of the most important browser-based protocols. Because normal authentication protocols are known to be prone to design errors, we expect that the additional restrictions of this protocol further complicate a secure design. The involvement of a standard Web browser indeed makes it very difficult to utilize prior research proposals such as [2, 1] for robust protocol design.

In general, we consider the SAML Single Sign-on protocol well-designed and carefully described. Nevertheless, further analysis of the protocol is necessary. The security aspects of the protocol are formulated in a constraint-based manner and structured according to the architecture of SAML. This is a common technique in this area, but can hamper a faultless implementation, because implementing software engineers may overlook a constraint or its impact on the protocol security. Furthermore, this kind of description complicates a general security analysis. Thus, the protocol description does not provide such an analysis, but an attack-by-attack description of countermeasures. This is a distinguishing feature to other protocols in this area, as some of them do not take such considerations at all. All told, the potential importance in industry and the new set of requirements make this protocol worth a closer look from a research perspective.

We present a general security analysis of the SAML Single Sign-on Browser/Artifact profile, which is the first one for this kind of protocol standard. We discovered security flaws, that allowed to several attacks on the protocol, some of them with possibly severe impact, such as man-in-the-middle attacks, attacks by information leakage, and message replay. We present these three attacks in detail and sketch further attack approaches.

The remainder of this paper is structured as follows: We present a short overview of the SAML message standard and its Single Sign-on profile in Section 2. Section 3 introduces related protocols and attacks on them as well as prior research in this area. In Section 4, we describe the communication types introduced by the SAML Single Sign-on protocol. Section 5 contains our model for the login procedure and subsequent user tracking. We describe the protocol schema of the SAML Single Sign-on Browser/Artifact profile in Section 6. In Section 7, we present three attacks on the protocol. We discuss the vulnerability of an implementation using SSL or TLS channels in Section 8, and conclude our analysis in Section 9.

2. Security Assertion Markup Language (SAML)

SAML is an open message standard that encodes security assertions and corresponding protocol messages in XML format. The message standard itself is described in [7]. SAML allows so-called protocol bindings [8] that embed SAML constructs in other structures for transport. SAML, for instance, builds on the Simple Object Access Protocol (SOAP) with its SOAP over HTTP binding. In addition, the SAML standard includes descriptions of the use of SAML assertions in communication protocols and frameworks [8]. These so-called profiles contain protocol flows and security constraints for applications of SAML.

The SAML Single Sign-on Browser/Artifact Profile describes the usage of SAML messages to perform a single sign-on operation involving three parties – a user $\mathcal{U}$ equipped with a standard browser $\mathcal{B}$, a source site $\mathcal{S}$, and a destination site $\mathcal{D}$. We depict the protocol flow in Figure 1.

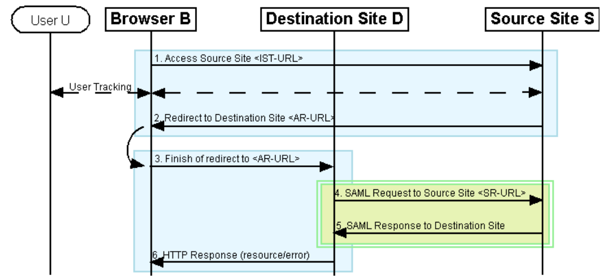

Figure 1. Protocol flow of the SAML Single Sign-on Browser/Artifact Profile.

The protocol assumes that user $\mathcal{U}$ authenticated itself to source site $\mathcal{S}$ beforehand. The protocol flow begins when user $\mathcal{U}$ returns to source site $\mathcal{S}$, for instance, having been redirected by a destination site $\mathcal{D}$. Source site $\mathcal{S}$ stores an assertion about the user's identity if it can recognize the browser $\mathcal{B}$ of user $\mathcal{U}$ during the so called user tracking. It then redirects the user's browser $\mathcal{B}$ to the destination site $\mathcal{D}$ the user wants to browse. Source site $\mathcal{S}$ includes a small piece of data, called a SAML artifact, into the redirect that refers to the assertion stored. Receiving the redirect with this artifact, destination site $\mathcal{D}$ shows this artifact to source site $\mathcal{S}$ and requests the corresponding assertion from it. By providing this assertion to $\mathcal{D}$, source site $\mathcal{S}$ confirms that user $\mathcal{U}$ presenting the SAML artifact was authenticated by $\mathcal{S}$.

3. Related Work

The first browser-based authentication protocol was, to our best knowledge, Microsoft Passport. Because the protocol is not published, we only refer to Microsoft's whitepapers such as [15]. Recent research discovered multiple vulnerabilities of Passport and described several severe attacks [12, 21]. A new attack was found at the beginning of May 2003, but no details have been published yet.

Two other projects base their protocols on the SAML message standard. The Liberty Alliance Project makes public proposals, but is not the subject of a standardization process [11]. One of its protocols using an enhanced client was vulnerable and attacked in [19]. The Shibboleth Project is a well-elaborated SAML application for inter-university federation [4]. In addition, [17] proposes a protocol called Browser-based Attribute Exchange (BBAE) that may also be built on SAML. This protocol concentrates on attribute exchange and privacy issues.

There are only few publications about general analyses of browser-based protocols. Recently, [18] provided an analysis of the privacy aspects of browser-based attribute-exchange protocols that covers SAML as well as Passport, Liberty and Shibboleth.

Considering prior research about general protocol design, two of the best-known publications are [1] and [2]. Whereas [1] suggests various informal principles about general protocol design, [2] extends them for the usage in public-key protocols. [10] presents more practical analyses of client authentication on the Web, which is distantly related to the browser-based authentication protocols. One of the most well-analyzed authentication protocols of research is the Needham-Schroeder protocol [16], which was analyzed and broken by [13] and [14]. These publications are useful as examples for protocol design and analysis, too.

4. Communication

The SAML Single Sign-on protocol constrains the methods used to transfer messages with several security properties. It names two combinations of such properties that we introduce in the following sections.

4.1. With Confidentiality and Integrity Only

The SAML Single Sign-on protocol refers to a message transfer that provides confidentiality and integrity but no authentication. Message transfers with these properties can be implemented by message security solutions or anonymous SSL/TLS channels [6]. This class of channels is naturally vulnerable to man-in-the-middle attacks. We formalize this kind of communication as follows:

$$\mathcal{S} \rightarrow_{cid} \mathcal{R}\text{: } adr - msg$$

The abstract identifiers $\mathcal{S}$ and $\mathcal{R}$ refer to the participating sender and recipient. We introduce the hostname adr and the message msg as parameters. Even if the message transfer does not utilize channels, we use a channel identifier cid that is written as an index of the send arrow. In an implementation without channels, cid represents the underlying network connection. The channel identifier is defined as the first message is sent. It can be given as input for further send operations to stress the fact that the messages are being transferred through the same connection. We omit the address adr in this case.

As the SAML Single Sign-on protocol does not use authentication in this kind of message transfer, we do not refer to fixed identities for the communicating parties. We therefore cannot relate the given security properties to such identities either. Because of the lack of this relation, it is nearly impossible to match the properties claimed to well-defined cryptographic properties. We describe the security properties in the following paragraph and divide integrity into verifiability and non-malleability.

Confidentiality: Apart from the original sender, only one party can decrypt the message msg. This will usually be the party that controls the host adr.

Integrity: *Verifiability* – A party that reads the message msg is able to verify whether msg is in its original state of the send operation. The identity of the sender cannot be verified.
Non-Malleability – Let $\mathcal{S}$ be an honest sender of a message msg to a recipient $\mathcal{R}$. Let an adversary $\mathcal{A}$ send messages msg'_i similar to msg to recipient $\mathcal{R}$. Then the messages $\mathcal{R}$ receives are either identical to msg or independent of it.

4.2. Secure Channels

The SAML Single Sign-on specifies a second kind of message transfer. It claims the security properties confidentiality, integrity and bilateral authentication. We interpret a transfer type with these properties as a secure channel. It can be implemented with SSL/TLS channels with bilateral authentication, i.e. with server- and client-side certificates. We use the following notation:

$$\mathcal{S}(snd\_id) \Rightarrow_{cid} \mathcal{R}(rcv\_id)\text{: } adr - msg$$

Again, we have two communicating parties, a sender $\mathcal{S}$ and a recipient $\mathcal{R}$, where $\mathcal{S}$ has an identity snd_id and $\mathcal{R}$ an identity rcv_id. We name the hostname of the recipient in the first send operation and omit it in subsequent steps if the same channel cid is used.

We describe the corresponding security properties in the following. We fix sender $\mathcal{S}$ and recipient $\mathcal{R}$ of a message msg to facilitate the formalization. Both participating parties can send and receive messages over an established channel.

Bilateral Authentication: Sender $\mathcal{S}$ and recipient $\mathcal{R}$ identify themselves with their identities snd_id and rcv_id. Both parties check the corresponding certificates of the communication partner. They only proceed with the protocol if there is a valid certificate chain to a trusted certification authority.

Confidentiality: Only sender $\mathcal{S}$ with identity snd_id and recipient $\mathcal{R}$ with identity rcv_id can read the message msg.

Integrity: The receiving $\mathcal{R}$ can verify whether the message msg was sent by a sender $\mathcal{S}$ with identity snd_id. If the recipient $\mathcal{R}$ receives a message msg, $\mathcal{R}$ either receives message msg in the state in which the server $\mathcal{S}$ sent it or gets an error message.

5. User Tracking

It is an important part of the SAML Single Sign-on protocol that source site $\mathcal{S}$ does not require user $\mathcal{U}$ to re-login, but is supposed to recognize $\mathcal{U}$ automatically. Thus, the entire protocol run can be accomplished without user interaction. We call the method for recognition of a user that logged in beforehand *user tracking*.

The protocol assumes that user $\mathcal{U}$ has already logged in earlier and that $\mathcal{U}$'s login has not timed out. When user $\mathcal{U}$'s browser $\mathcal{B}$ is redirected back to source site $\mathcal{S}$ in protocol step 1, $\mathcal{S}$ is able to link the browser $\mathcal{B}$ to the still valid login. Source site $\mathcal{S}$ deduces the identity of user $\mathcal{U}$ from this link. We formulate the login itself as follows:

(a) $\mathcal{S} \rightarrow_{cid} \mathcal{B} \rightsquigarrow U$: login request

(b) $\mathcal{U} \rightsquigarrow \mathcal{B} \rightarrow_{cid} \mathcal{S}$: login $l_{\mathcal{U},\mathcal{S}}$

(c) $\mathcal{S} \rightarrow_{cid} \mathcal{B}$: verification information vi

In the login step, source site $\mathcal{S}$ initiates the user authentication using a given channel represented by its channel identifier cid. Browser $\mathcal{B}$ presents a login request to user $\mathcal{U}$, which we denote with the leads-to ($\rightsquigarrow$) symbol. In the second step, user $\mathcal{U}$ inputs its login information $l_{\mathcal{U},\mathcal{S}}$ into browser $\mathcal{B}$. This login information can for instance be a combination of user $\mathcal{U}$'s username and password. Browser $\mathcal{B}$ forwards it to source site $\mathcal{S}$ through the channel referenced by cid. Source site $\mathcal{S}$ resolves $\mathcal{U}$'s login information $l_{\mathcal{U},\mathcal{S}}$ to its identity $id_{\mathcal{U}}$. After a successful login, source site $\mathcal{S}$ sends a piece of verification information vi back to browser $\mathcal{B}$. This piece of information later confirms user $\mathcal{U}$'s login.

The subsequent user tracking works without further user interaction:

(d) $\mathcal{S} \rightarrow_{cid'} \mathcal{B}$: request for vi

(e) $\mathcal{B} \rightarrow_{cid'} \mathcal{S}$: proof of knowledge of vi

Source site $\mathcal{S}$ initiates the *user tracking* and requests the proof of knowledge of the verification information vi through a given channel with id cid'. The browser looks up the verification information and responds to $\mathcal{S}$. Having been convinced of the browser $\mathcal{B}$'s knowledge, the source site resolves user $\mathcal{U}$'s identity $id_{\mathcal{U}}$.

6. Protocol Schema

In this section we describe the protocol schema of the SAML Single Sign-on Browser/Artifact profile in detail. We already introduced the protocol run in Section 2 and depicted the protocol flow in Figure 1.

6.1. Step 1: Contact the Source Site

In the first protocol step, browser $\mathcal{B}$ establishes contact to source site $\mathcal{S}$.

(a) $\mathcal{B} \rightarrow_{bs\_cid} \mathcal{S}$: ist_host – GET <IST-URL>$_{\mathcal{S}}$? TARGET=$target$ … <HTTP-Version>

(b) $\mathcal{S}$: Checks the request.

Browser $\mathcal{B}$ connects to the inter-site transfer URL <IST-URL>$_{\mathcal{S}}$ of $\mathcal{S}$. $\mathcal{B}$ and $\mathcal{S}$ use the message transfer with confidentiality and integrity as described in Section 4.1 to send a target description $target$ to $\mathcal{S}$. In substep 1b, source site $\mathcal{S}$ parses the request and initiates a new session for it. $\mathcal{S}$ tries to extract a target description $target$ from the request's query string.

Lack of Authentication: This connection provides no unilateral authentication. Thus, browser $\mathcal{B}$ cannot necessarily verify the identity of $\mathcal{S}$ by checking the server certificate of $\mathcal{S}$ and the certificate chain to a trusted certification authority. This lack of certification is a cornerstone of man-in-the-middle attacks on the communication between browser and source site ($\mathcal{B} \leftrightarrow \mathcal{A} \leftrightarrow \mathcal{S}$). An adversary who wants to act as man-in-the-middle at this point only needs to break the user tracking in the subsequent step to succeed.

Message Format: The message sent to $\mathcal{S}$ is an HTTP GET request for the path of <IST-URL>$_{\mathcal{S}}$. It contains the target resource $target$ that user $\mathcal{U}$ wants to access on destination site $\mathcal{D}$. The SAML Single Sign-on Profile neither specifies further elements of this URL nor does it prohibit the inclusion of further elements. The protocol description lacks an explicit naming of protocol type or step, which hampers the dispatching of messages to different protocol modules. This also allows a cumulation of artifacts within this URL by a malicious destination site and repetition of the protocol steps 1 to 3.

User Tracking

As source site $\mathcal{S}$ holds a connection to the browser at this time, it is a canonical solution to use this connection for the user tracking. Unfortunately this connection does not provide authentication according to the SAML Single Sign-on Profile. Therefore verification information cannot be bound to a certified identity. We assume that browser $\mathcal{B}$ stores it by the hostname of source site $\mathcal{S}$.

(a) $\mathcal{S} \rightarrow_{bs\_cid} \mathcal{B}$: request for vi

(b) $\mathcal{B} \rightarrow_{bs\_cid} \mathcal{S}$: knowledge of vi

In the situation described a man-in-the-middle can forward the communication between $\mathcal{B}$ and $\mathcal{S}$. The two honest parties cannot distinguish the adversary from the intended communication partner. In general, it is a design flaw that the possibility of a man-in-the-middle attack is dependent on a strong user tracking. As user tracking is unspecified in the SAML Single Sign-on protocol, we cannot assume that it is resistant against such attacks.

6.2. Step 2: Initiating the Redirect to the Destination Site

After a successful recognition of user $\mathcal{U}$, source site $\mathcal{S}$ generates one or more SAML artifacts and includes them in a so-called SAML searchpart. It redirects browser $\mathcal{B}$ to the artifact receiver URL <AR-URL>$_\mathcal{D}$ of destination site $\mathcal{D}$ with the SAML searchpart as query string.

(a) $\mathcal{S}$: Determines the destination site $\mathcal{D}$ corresponding to the $target$ of Step 1. Looks up <AR-URL>$_\mathcal{D}$ in its artifact receiver table.

(b) $\mathcal{S}$: Generates one or more SAML artifacts <SAMLart>$_i$ that contain its $SourceID_\mathcal{S}$.

(c) $\mathcal{S}$: Generates a SAML searchpart $SAMLsp$ that contains the target description and the generated artifacts.

(d) $\mathcal{S} \rightarrow_{bs\_cid} \mathcal{B}$: <HTTP-Version> 302 <Reason Phrase>
Location <AR-URL>$_\mathcal{D}$? $SAMLsp$

In substep 2a, the source site looks up the artifact receiver URL <AR-URL>$_\mathcal{D}$. As the SAML Single Sign-on protocol does not specify this procedure, we assume that it searches for an artifact receiver URL having a hostname equal to that of the target resource. The source site generates a number of SAML artifacts and stores the information that the artifacts were issued to destination site $\mathcal{D}$. As the source site does not have a certificate or identity of $\mathcal{D}$, it can only use destination site $\mathcal{D}$'s hostname or <AR-URL>$_\mathcal{D}$ as reference.

In this step we experience the same problems with the message format as in Step 1. Moreover, as we cannot rely on authentication either, the man-in-the-middle attack between $\mathcal{B}$ and $\mathcal{S}$ is possible.

6.3. Step 3: Redirect to the Destination Site

In Step 3, browser $\mathcal{B}$ connects to destination site $\mathcal{D}$. $\mathcal{B}$ delivers the SAML artifacts to the <AR-URL>$_\mathcal{D}$.

(a) $\mathcal{B}$: Extracts the URL <AR-URL>$_\mathcal{D}$? $SAMLsp$ from the Location String of step 2 response.

(b) $\mathcal{B} \rightarrow_{bd\_cid} \mathcal{D}$: ar_host – GET <AR-URL>$_\mathcal{D}$? $SAMLsp$ <HTTP-Version>

Lack of Authentication: In this protocol step we again do not have unilateral authentication. Because the protocol does not specify short-term freshness measures or the necessity of channel-based security, a replay attack may be possible. An adversary can replay a message-wise encrypted HTTP request to the <AR-URL>$_\mathcal{D}$. The lack of authentication also allows a man-in-the-middle attack between $\mathcal{B}$ and $\mathcal{D}$. This is critical as the SAML Single Sign-on protocol trusts the assumption that $\mathcal{D}$ is connected to browser $\mathcal{B}$ of user $\mathcal{U}$.

6.4. Step 4: SAML Request

In step 4 the destination site establishes a secure channel to source site $\mathcal{S}$. It uses the $SourceID$ of the SAML artifacts received to find the corresponding SAML responder URL and sends a SAML request to it.

(a) $\mathcal{D}$: Check that each artifact <SAMLart>$_i$ the $SAMLsp$ contains the same $SourceID$.

(b) $\mathcal{D}$: Look up <SR-URL>$_\mathcal{S}$ corresponding to $SourceID$

(c) Generate RequestID.

(d) $\mathcal{D}(id_\mathcal{D}) \Rightarrow_{ds\_cid} \mathcal{S}(id_\mathcal{S})$: sr_host – SAML request to <SR-URL>$_\mathcal{S}$ containing the artifacts <SAMLart>$_i$

In step 4a, the destination site generates a session for the incoming request and analyzes the URL of the request. The destination site aborts the protocol run if the request is not intended for the <AR-URL>$_\mathcal{D}$, if $\mathcal{D}$ cannot parse the SAML searchpart, or if the request does not include SAML artifacts. The destination site checks the validity of the artifacts <SAMLart>$_i$ in the searchpart. All artifacts must be well formated, have a valid version id, and contain the same non-empty $SourceID$. Destination site $\mathcal{D}$ uses this $SourceID$ to look up the <SR-URL>$_\mathcal{S}$ of source site $\mathcal{S}$ (substep 4b). In substep 4d, the destination site establishes a secure channel ds_cid to source site $\mathcal{S}$ with bilateral authentication. It sends a SAML request with the received artifacts through this channel.

Specification of the Source Site Lookup: The protocol step does not fully specify which information destination site $\mathcal{D}$ knows of source site $\mathcal{S}$. The SAML Single Sign-on protocol only states that destination site $\mathcal{D}$ can look up <SR-URL>$_\mathcal{S}$. Thus, we can assume that $\mathcal{D}$ cannot look up the identity $id_\mathcal{S}$ of $\mathcal{S}$.

One-request Property of the SAML Artifact: According to the SAML Single Sign-on protocol, SAML artifacts can be used only once. This is enforced by the source site in step 5, while destination site $\mathcal{D}$ does not store the artifacts received. Thus, if the transfer of the SAML artifacts to the source site fails, the artifacts are still valid and reusable. As the SAML Single Sign-on protocol specifies no resend operation upon such a failure, it may leave free valid artifacts.

6.5. Step 5: SAML Response

In Step 5, the source site analyzes the SAML request and generates an adequate response. It sends the response back through the channel with id ds_cid generated in step 4.

(a) $\mathcal{S}$: Check for artifact destination equality.

(b) $\mathcal{S}$: Enforce of the one-time use of the artifacts.

(c) $\mathcal{S}$: Look up or generate SAML assertions corresponding to the artifacts $<$SAMLart$>_i$.

(d) $\mathcal{S}$: Generate ResponseID.

(e) $\mathcal{S}(id_\mathcal{S}) \Rightarrow_{ds\_cid} \mathcal{D}(id_\mathcal{D})$: SAML response from $<$SR-URL$>_\mathcal{S}$ containing SSO assertions about $id_\mathcal{U}$ or an error code. The response references the $RequestID$ of the SAML request of step 4 in the $InResponseTo$ element.

(f) $\mathcal{D}$: Verify validity of SAML response.

In substep 5a, source site $\mathcal{S}$ checks whether the received artifacts were submitted from the same destination site, to which it issued them. According to the SAML Single Sign-on profile, the source site must return a response with no assertions if the artifacts were issued to another destination site. As the source site $\mathcal{S}$ can only have stored the hostname or $<$AR-URL$>$ to which it issued the artifacts, it will typically only compare the destination site's hostname with the stored hostname entry. If in substep 5b any of the artifacts have already been seen, source site $\mathcal{S}$ responds with an empty response. This is called the one-time request property of the SAML artifact. Each artifact can be used only once. Then in substep 5c the source site tries to determine the assertions corresponding to the artifacts given. If one of the assertions cannot be generated or looked up, $\mathcal{S}$ responds with no assertions. Source site $\mathcal{S}$ generates a SAML response corresponding to the request id of the request given, and sends the SAML response back through the channel generated in step 4.

Destination site $\mathcal{D}$ checks the response received in the subsequent substep. It checks whether the $InResponseTo$ element is equal to its own $RequestID$. Then it verifies the validity of the SAML assertions. There must be at least one SAML SSO assertion included in the response, and there must be exactly one assertion for each submitted SAML artifact. If these checks fail, $\mathcal{D}$ aborts the protocol run. Otherwise $\mathcal{D}$ analyzes the SAML assertions to determine $\mathcal{U}$'s identity $id_\mathcal{U}$.

One-request Property of the SAML Artifact: The SAML Single Sign-on protocol states a one-time usage constraint for the SAML artifacts: If a SAML artifact is presented to the source site again, the source site must return the same message it uses upon a query with an unknown artifact. This property must be implemented with care as otherwise there is the danger of race conditions or left over valid artifacts because of protocol failures.

Multiple Services on One Host: A source site $\mathcal{S}$ can only verify that it issued the artifact to the same hostname that submitted it in step 4. If there are multiple services on destination site $\mathcal{D}$, a malicious low-security service could use received artifacts to impersonate user $\mathcal{U}$ to a high-security service. Let us consider this example: A bank $\mathcal{D}_{Bank}$ hosts a stock-market analysis toolkit $stock$ and its e-banking application $ebank$ on the same host. It uses the SAML Single Sign-on for customization of the analysis toolkit and for login to user $\mathcal{U}$'s e-banking account. A malicious stock-market analysis tool $stock'$ could forward the artifacts received at $<$AR-URL$>_{\mathcal{D}_{Bank},stock'}$ to $<$AR-URL$>_{\mathcal{D}_{Bank},ebank}$ and make e-banking transactions with the permissions of user $\mathcal{U}$.

6.6. Step 6: Response to the Browser

In step 6, destination site $\mathcal{D}$ responds to browser $\mathcal{B}$'s request. If $\mathcal{D}$ is convinced of user $\mathcal{U}$'s identity $id_\mathcal{U}$, it will present the target resource requested. Otherwise it will typically reply with an error message.

(a) $\mathcal{D} \rightarrow_{bd\_cid?} \mathcal{B}$: resource page $target$ or error message.

Specification of this Step: The SAML Single Sign-on protocol does not specify step 6 precisely: It leaves unspecified which connection to use and claims no security properties for this connection. We assume that the destination site uses the connection of step 3 identified by bd_cid to respond to the browser. We also assume that it has the same security properties as in step 3. This assumption is justified because HTTP servers normally respond through the same connection through which they received a request.

HTTP Referrer Tag: The SAML Single Sign-on protocol does not specify the details of the HTTP message either. Therefore we can assume that the protocol does not handle the referrer tag of HTTP. According to [9] it may be set if the referrer has its own URI. This is the case for the $<$AR-URL$>_\mathcal{D}$ to which the browser is connected. As the referrer tag includes the query string, it also contains the artifacts previously sent to $\mathcal{D}$. Thus, if artifacts were not consumed by source site $\mathcal{S}$, an adversary could obtain valid outstanding artifacts.

7. Attacks

We present three attacks on the SAML Single Sign-on Browser/Artifact profile that are based on the flaws identified above.

7.1. Connection Hijacking / Replay Attack

An adversary can break the SAML Single Sign-on Profile by connection hijacking and replay of an encrypted redirect. The technique of connection hijacking is described in [5]. The prerequisites for this are as follows:

Prerequisites: An adversary $\mathcal{A}$ is capable of connection interception and can observe the connection from browser $\mathcal{B}$ to artifact receiver URL $<$AR-URL$>_{\mathcal{D}}$ of Step 3. In addition, the integrity property claimed in Step 3 is interpreted as message integrity without binding to the sending party and replay prevention.

The Attack: The attack refers to the protocol schema in Section 6. The attacker intercepts the step 3 redirect to destination site $\mathcal{D}$ and replays the encrypted message.

3. $\mathcal{B} \rightarrow \mathcal{D}$: Redirect p to $<$AR-URL$>_{\mathcal{D}}$
$\mathcal{A}$: Adversary $\mathcal{A}$ intercepts this redirect and finishes the connection from $\mathcal{B}$ to $\mathcal{D}$.

3*. $\mathcal{A}_{\mathcal{B}} \rightarrow \mathcal{D}$: replay of redirect p to $<$AR-URL$>_{\mathcal{D}}$
The adversary resends the message as seen in the preceding step, impersonating browser $\mathcal{B}$ to $\mathcal{D}$. It utilizes a modified channel subprotocol that allows already encrypted messages to be sent directly.
The destination site cannot distinguish between $\mathcal{B}$ and $\mathcal{A}$ because of the lack of authentication. $\mathcal{D}$ therefore proceeds as specified in the protocol.

4. $\mathcal{D} \rightarrow \mathcal{S}$: SAML request q with SAML artifacts of p.

5. $\mathcal{S} \rightarrow \mathcal{D}$: SAML response r with assertions.
The artifacts given in the SAML request q of $\mathcal{D}$ were issued to $\mathcal{D}$. The SAML request is identical to the one $\mathcal{D}$ would have sent if it were connected to $\mathcal{B}$.

6*. $\mathcal{D} \rightarrow \mathcal{A}_{\mathcal{B}}$: Response to Step 3*
$\mathcal{D}$ will respond to $\mathcal{A}_{\mathcal{B}}$ and grant $\mathcal{A}_{\mathcal{B}}$ the same permissions as $\mathcal{B}$ with user $\mathcal{U}$ and certified identity $id_{\mathcal{U}}$.

Discussion: Because of the integrity and confidentiality property of step 3, adversary $\mathcal{A}$ cannot see or modify the content of the message p. But these properties are not strong enough to prevent a replay attack. Adversary $\mathcal{A}$ impersonates browser $\mathcal{B}$ to destination site $\mathcal{D}$. Because of the lack of authentication in this step and missing identifiers in the redirect, $\mathcal{D}$ cannot distinguish between the parties $\mathcal{B}$ and $\mathcal{A}_{\mathcal{B}}$. Apart from the IP address of the communication partner, the view of $\mathcal{D}$ is the same in the communication with $\mathcal{B}$ and $\mathcal{A}_{\mathcal{B}}$.

Possible solutions: Source site $\mathcal{S}$ can input the IP address of browser $\mathcal{B}$ as query string parameter into the redirect. The destination site $\mathcal{D}$ checks the IP address in the message received for equality with the IP address of browser $\mathcal{B}$. If this check fails, $\mathcal{D}$ aborts the protocol run. This measure might interfere with the IP rollover of certain ISPs, because it produces a false positive if $\mathcal{B}$'s IP address is changed between steps 2 and 3. A second possibility to prevent the described attack is to enhance the integrity property claimed for the steps 1-3 so that it includes binding to the sending party or the underlying channel. One can also use a secure channel $\mathcal{B} \leftrightarrow \mathcal{D}$ in steps 3 and 6, which provides freshness and replay prevention.

7.2. Man-in-the-Middle Attacks

In this section we consider a concrete man-in-the-middle attack. We describe further techniques and entry points for this kind of attack.

The general technique of man-in-the-middle attacks is described in [3]. [14] provides a concrete example of such an attack on the Needham-Schroeder Public-Key protocol.

7.2.1. Between $\mathcal{B}$ and $\mathcal{S}$ by DNS Spoofing This attack uses the well-known weakness that an adversary who controls the Domain Name Service (DNS) can impersonate one party to another. We present a man-in-the-middle attack in which adversary $\mathcal{A}$ is a proxy between browser $\mathcal{B}$ and source site $\mathcal{S}$: $\mathcal{B} \leftrightarrow \mathcal{A} \leftrightarrow \mathcal{S}$.

Prerequisites: Let $\mathcal{A}$ be an adversary that can break DNS. By doing so, the adversary can act with a hostname not belonging to him or her and can impersonate certain URLs to other parties. We additionally assume that the method of tracking an authenticated user of source site $\mathcal{S}$ is unprotected against man-in-the-middle attacks.

The Attack: Adversary $\mathcal{A}$ uses its ability to break DNS to impersonate the inter-site transfer URL $<$IST-URL$>_{\mathcal{S}}$ of source site $\mathcal{S}$ to browser $\mathcal{B}$. $\mathcal{A}$ forwards all communication until it obtains of unused SAML artifacts.

$\mathcal{A}_{\mathcal{S}}$: Impersonates the source site $\mathcal{S}$'s <IST-URL>$_{\mathcal{S}}$ to browser $\mathcal{B}$. $\Longrightarrow$ <IST-URL>$_{\mathcal{S}}^{(\mathcal{B})}$=<IST-URL>$_{\mathcal{A}}$.

1. $\mathcal{B} \rightarrow \mathcal{A}_{\mathcal{S}}$: finish of the redirect
Note that the profile does not claim authentication for this step.

1*. $\mathcal{A}_{\mathcal{B}} \rightarrow \mathcal{S}$: finish of the redirect
The adversary $\mathcal{A}_{\mathcal{B}}$ impersonates browser $\mathcal{B}$ to $\mathcal{S}$.

$\mathcal{A}$: Acts as man-in-the-middle between $\mathcal{B}$ and $\mathcal{S}$. Because the user-tracking system of $\mathcal{S}$ is not resistant against man-in-the-middle attacks, $\mathcal{A}$ can forward all communication between $\mathcal{B}$ and $\mathcal{S}$ during the user tracking.

2*. $\mathcal{S} \rightarrow \mathcal{A}_{\mathcal{B}}$: redirect request to <AR-URL>$_{\mathcal{D}}$
This redirect contains the SAML artifacts for $\mathcal{B}$ readable by $\mathcal{A}_{\mathcal{B}}$. $\mathcal{A}_{\mathcal{B}}$ now begins to impersonate $\mathcal{B}$ to destination site $\mathcal{D}$.

3*. $\mathcal{A}_{\mathcal{B}} \rightarrow \mathcal{D}$: finish of redirect to <AR-URL>$_{\mathcal{D}}$
The adversary $\mathcal{A}_{\mathcal{B}}$ impersonates $\mathcal{B}$ to $\mathcal{D}$. The redirect contains valid SAML artifacts and allows $\mathcal{A}_{\mathcal{B}}$ to act using the permissions of $\mathcal{U}$.

2. $\mathcal{A} \rightarrow \mathcal{B}$: redirect request to <IST-URL>$_{\mathcal{S}}$
Adversary $\mathcal{A}$ sends the original redirect to browser $\mathcal{B}$. As the profile assumes that the user has already been authenticated to $\mathcal{S}$, there need not be any user interaction between protocol steps 1 and 2. Therefore, the adversary can re-initiate a normal protocol run of $\mathcal{B}$ with a high probability that the user will not notice the reset of the protocol run.

Discussion: The profile does not claim unilateral authentication in steps 1 and 2. Therefore browser $\mathcal{B}$ cannot distinguish between the <IST-URL>$_{\mathcal{A}}$ of adversary $\mathcal{A}_{\mathcal{S}}$ and <IST-URL>$_{\mathcal{S}}$ that belongs to the honest source site $\mathcal{S}$. The profile does not state which security assumptions can be made about the tracking of an authenticated user by source site $\mathcal{S}$. Therefore we can assume that the adversary can forward this communication as well.

Possible solutions: Unilateral authentication in all protocol steps can prevent the adversary from impersonating source site $\mathcal{S}$ to browser $\mathcal{B}$. Strong user tracking, resistant against man-in-the-middle attacks, can also help. However, as the user tracking is not specified in the SAML Single Sign-on protocol, it is not safe to rely on it.

7.2.2. Other Man-in-the-Middle Attacks To accomplish the described man-in-the-middle attack in a non-portal scenario, an adversary can also manipulate the target of the step-1 redirect. It can rewrite the HTTP response that initiates the redirect and change the target URL. As the SAML Single Sign-on protocol specifies no security properties for this step 0, we can assume that the connection is unsecured.

An adversary also has the option to act as man-in-the-middle between browser $\mathcal{B}$ and destination site $\mathcal{D}$. As there is no unilateral authentication required in steps 3 and 6, a browser $\mathcal{B}$ cannot distinguish between an adversary $\mathcal{A}_{\mathcal{D}}$ and destination site $\mathcal{D}$.

7.3. HTTP Referrer Attack

The following attack allows an adversary $\mathcal{A}$ to provoke an information leakage of valid SAML artifacts. It uses the Referrer Tag of HTTP (see [9]) to obtain unused SAML artifacts.

Prerequisites: To accomplish this attack we need an adversary $\mathcal{A}$ that can tap channels and intercept arbitrary connections. In addition, let $\mathcal{B}$ be a browser that sets the HTTP Referrer Tag by default.

We require some properties of the error messages produced by $\mathcal{D}$ if a protocol failure occurs: either the error page contains a link to a URL not providing confidentiality or integrity, or adversary $\mathcal{A}$ is able to manipulate data transferred through connections that do not maintain the integrity property.

The Attack: The attack refers to the protocol schema in Section 6. The adversary provokes an information leakage of valid unused artifacts by interrupting the connection between destination site $\mathcal{D}$ and source site $\mathcal{S}$.

3. $\mathcal{B} \rightarrow \mathcal{D}$: redirect to <AR-URL>$_{\mathcal{D}}$
which contains valid SAML artifacts.

4. $\mathcal{D} \rightarrow \mathcal{S}$: SAML request to <SR-URL>$_{\mathcal{S}}$
$\mathcal{A}$: Adversary $\mathcal{A}$ intercepts this message. Therefore the SAML request is unsuccessful, and $\mathcal{D}$ sends an error message in the HTTP response of step 6.

6. $\mathcal{D} \rightarrow \mathcal{B}$: error message
According to the profile no confidentiality or integrity is required in this and subsequent steps. The adversary $\mathcal{A}$ can proceed in two ways:
   - If the error message contains a link or redirect to a URL to party $\mathcal{P}$ that is not secured, $\mathcal{A}$ can tap the channel the browser possibly establishes to $\mathcal{P}$.
   - If the adversary $\mathcal{A}$ is able to manipulate data transferred through connections, he can change the message of step 6 to a redirect to a party $\mathcal{P}$ it has chosen.

7. $\mathcal{B} \rightarrow \mathcal{P}$: HTTP request
The next request of the browser $\mathcal{B}$. It contains <AR-URL>$_{\mathcal{D}}$ in the referrer tag including the query string and therefore the still valid SAML artifacts of step 4. $\mathcal{A}$ can read them in plain text.

3*. $\mathcal{A} \rightarrow \mathcal{D}$: redirect to $<$AR-URL$>_{\mathcal{D}}$
as read in the referrer tag.

Discussion: The message of step 6 allows information leakage: It does not claim confidentiality or integrity. The referrer tag is normally set if the source from which the user comes, has its own URI. It will contain the $<$AR-URL$>_{\mathcal{D}}$, including the query string that contains the still valid SAML artifacts.

Possible solutions: The HTTP specification [9] recommends not including the Referer tag in a request if the referring page was transferred with a secure protocol. Still, the implementation of this recommendation by Web browsers is beyond of the sphere of influence of SAML implementers. We can also prevent this attack by using a dereferrer redirect before step 6. This is a common technique in current e-commerce implementations. This redirect is only needed if steps 4 or 5 were not successful or not all artifacts were consumed by $\mathcal{D}$. A second approach is to enforce the one-time usage property of the SAML artifacts also at the destination site. As destination site $\mathcal{D}$ has already seen the artifacts in the regular step 3, it will not accept them a second time. As source site $\mathcal{S}$ checks whether received artifacts were issued to the sending destination site, the artifacts cannot be used in the communication with other destination sites.

8. Vulnerability of the SSL/TLS Binding

SOAP over HTTP is one of the most important bindings of the SAML Single Sign-on protocol. It utilizes SSL 3.0 or TLS 1.0 with unilateral authentication as communication channel for connections that require confidentiality and integrity.

As this binding exceeds the security requirements of the protocol itself, our attacks will be more difficult to accomplish. The replay attack described in Section 7.1 no longer works, because the channels provide replay prevention. For the man-in-the-middle attack of Section 7.2 a stronger adversary is needed. This adversary not only needs to impersonate $<$IST-URL$>_{\mathcal{S}}$ but also present a valid certificate. As the user tracking itself does not require user interaction, the user has little chance to verify the certificate of source site $\mathcal{S}$. The referrer attack of Section 7.3 still works, as the SAML Single Sign-on protocol does not require any security measures in step 6 and therefore no SSL/TLS channel is used. Attacks in which one has several services on the same destination site and one service holder tries to cheat another are still possible. They do not depend on the insecurity of the communication channels. The same holds for attacks that are based on race conditions and try to double-spend artifacts.

In general, it is a good idea to use SSL 3.0 or TLS 1.0 as communication channel. Their use enhances the security of the SAML Single Sign-on protocol dramatically, but does not guarantee complete security.

9. Conclusion

We have described several design flaws of the SAML Single Sign-on Browser/Artifact profile. In several instances, the protocol description is imprecise. We presented three attacks on the protocol to demonstrate its vulnerability. Even a protocol binding with underlying SSL/TLS channels and unilateral authentication can be broken. Most implementations will simply use SSL/TLS channels with unilateral authentication, which complicates or prevents man-in-the-middle and replay attacks. This leaves the discovered referrer attack and attacks where the destination site provides multiple services on the same host.

We have deduced several recommendations for the design of browser-based protocols from our analysis. First of all, we strongly recommend that secure channels such as SSL 3.0 or TLS 1.0 with unilateral authentication for message transfer always be used. They outmatch normal transfer of signed and encrypted messages, as they provide authentication, freshness, and replay prevention. We also recommend including more explicitness measures into the messages. It is important to name protocol type, protocol step, source and destination of a message explicitly in the message. Such measures could for instance prevent attacks where multiple services of a site are involved. We recommend not only considering successful protocol runs, but also analyzing all states the protocol can reach. Especially error states may hide opportunities for attacks such as our referrer attack.

We are convinced that the SAML Single Sign-on Browser/Artifact profile is in general a well-written protocol. In fact, it is one of the most carefully designed browser-based protocols in federated identity management. Nevertheless, several changes are required to improve its security and prepare for its broad application in industry.

Acknowledgment

We thank Michael Backes, Charlotte Bolliger, Günter Karjoth, Lilli-Marie Pavka, and Birgit Pfitzmann for the valuable reviews and helpful discussions.

References

[1] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, 1996.

[2] R. Anderson and R. Needham. Robustness principles for public key protocols. In *CRYPTO: Proceedings of Crypto*, pages 236–247, Berlin, 1995. Springer-Verlag.

[3] B. B. Bhansali. Man-in-the-middle attack - a brief, February 2001.

[4] S. Cantor and M. Erdos. Shibboleth-architecture draft v05, May 2002.

[5] P. Dave and N. Moussa. TCP connection hijacking, 2002.

[6] T. Dierks and C. Allen. RFC 2246: The TLS protocol, January 1999. Status: Standards Track.

[7] P. H.-B. et al. Assertions and protocol for the OASIS security assertion markup language (SAML), 2002.

[8] P. M. et al. Bindings and profiles for the OASIS security assertion markup language (SAML), 2002.

[9] R. T. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC 2616: Hypertext transfer protocol – HTTP/1.1, June 1999. Status: Standards Track.

[10] K. Fu, E. Sit, K. Smith, and N. Feamster. Dos and don'ts of client authentication on the web. In *Proceedings of the 10th USENIX Security Symposium*, 2001.

[11] J. Hodges and T. Wason. Liberty architecture overview, 2003.

[12] D. P. Kormann and A. D. Rubin. Risks of the passport single signon protocol. *Computer Networks*, 33:51–58, 2000.

[13] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055, pages 147–166. Springer-Verlag, Berlin Germany, 1996.

[14] C. Meadows. Analyzing the needham-schroeder public-key protocol: A comparison of two approaches. In *ESORICS: European Symposium on Research in Computer Security*. LNCS, Springer-Verlag, 1996.

[15] Microsoft. .net passport review guide, 2002.

[16] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):393–399, 1978.

[17] B. Pfitzmann and M. Waidner. BBAE – a general protocol for browser-based attribute exchange. Research report RZ 3455 (# 93800), IBM Research Division, Zurich, June 2002.

[18] B. Pfitzmann and M. Waidner. Privacy in browser-based attribute exchange. In *Proceeding of the ACM Workshop on Privacy in the Electronic Society*, pages 52–62, Washington, DC, 2002. ACM Press.

[19] B. Pfitzmann and M. Waidner. Token-based web single signon with enabled clients. Technical Report IBM Research Report RZ 3458, IBM Research Division, 2002.

[20] J. Rouault and T. Wason. Liberty bindings and profiles specification, 2003.

[21] M. Slemko. Microsoft passport to trouble, 2001.