Semantic Insecurity: Security and the Semantic Web

Harry Halpin

INRIA
2 rue Simone Iff 75012 Paris, France
harry.halpin@inria.fr

Abstract. Strangely enough, the Semantic Web has fallen behind the rest of the Web in terms of security. TLS is not in use currently for the majority of URIs on the Semantic Web, and this leads to a number of attacks on the network, web, and even semantic level. After explaining the necessity and arguments over the TLS on the Semantic Web, we point out security and privacy flaws in WebID+TLS, the recent standards from the W3C Social Web Working Group. We propose a new kind of attacker, a semantic attacker, that attacks inference procedures. Lastly, we propose alternatives, including the use of modern cryptography that prevent attacks by virtue of using TLS, and how W3C standards such as the W3C Web Cryptography API and IETF OAuth can solve the use-cases needed by the Semantic Web.

Keywords: security, TLS, WebID, Semantic Web, RDF

1 Introduction

The Semantic Web is based on linked data where items of interest are identified by URIs (Uniform Resource Identifiers) such as *http://www.example.org*. Due to this design choice, to a large extent the Semantic Web crucially depends on these identifiers being able to successfully retrieve documents in order to put the "Web" into the Semantic Web. Strangely enough, the Semantic Web has fallen behind the rest of the Web in terms of security. As the core Semantic Web technologies were standardized by the W3C before the widespread use of TLS and the Same Origin Policy was formalized, the Semantic Web was designed without any security considerations. Yet today there is almost no academic work on security in terms of the Semantic Web [23]. Rather unfortunately, there also seems to be considerable confusion about security in existing Semantic Web deployments like Solid,[1] ranging from confusion over the security problems in HTTP URIs to misuse of cryptography in WebID+TLS. This also problematically leads to the Semantic Web to become a privacy risk for personal data, rather than being a force for privacy.

[1] https://solid.mit.edu

First, we review related literature on security, showing how this is the first work to look at the security of the Semantic Web itself in Section 2. Then security is defined in Section 3 via the classical definition of semantic security from provable security, and we introduce the symbolic model for protocol analysis [14]. Then we outline the three different kinds of attacks on Semantic Web architecture, following the lead of Barth et al.'s formalization of Web Security in general, but with the addition of a new attacker, the *semantic* attacker [2]:

1. The **Network attacker**: On the network level, we show TLS is not in use currently for the majority of URIs on the Semantic Web, leading to trivial attacks on Linked Data in Section 4.
2. The **Web attacker**: On the level of Web applications, we show proposed standards like WebID+TLS and the W3C Social Web standards have cryptographic security flaws in Section 5.
3. The **Semantic attacker**: On the level of inference procedures, we show how the preceding two levels can lead to attacks that can lead to corrupted inferences in Section 6

In general, the dependency of data retrieval and inferences based on insecure Semantic Web data can lead to attacks on trusted semantics of the Semantic Web itself. In Section 7, we demonstrate that this does not have to be the case: A number of standards from the IETF and W3C can be used to upgrade the Semantic Web to modern security-best practices, leading to a secure Semantic Web.

2 Related Literature

There has been a large amount of previous research on the security of the Semantic Web, yet none of it looks at the security properties of the Semantic Web infrastructure itself. In general, there have been three streams of research on security on the Semantic Web: Semantic Web policy languages for access control, Semantic Web ontologies for cybersecurity, and problems with privacy in publishing Semantic Web data.

2.1 Semantic Web-based Access Control

By far the largest amount of research has been done on access control languages for the Semantic Web, such as Rei [20] and more recently the use of N3Logic [5] with a sketch of a future W3C standard for Web Access Control (WAC).[2] All of these approaches descend from an attempt to model the permissions and obligations between agents that would be operating on Semantic Web data using speech acts [20], with the goal to maintain some security policy during a distributed activity such as Semantic Web Service composition. In practice these approaches are mostly useful for interoperation between various access control

[2] https://www.w3.org/wiki/WebAccessControl

policies. Policy languages are typically used with mandatory access control (possibly extended to role-based and attribute-based access control), which relies on a central trusted party such as an OS to actually enforce the access control [25], and such a centralized trusted party is missing on the open Web except on the level of individual web-sites. The primary innovation in using the Semantic Web for policy languages is distributed access control. Rather than policy languages, much of the rest of the Web depends on a capabilities-based approach as given by OAuth [15]. This line of research on policy languages has been the source of much innovation but is orthogonal to the security concerns of Semantic Web data itself, upon which these policy languages depend for use in their inference.

2.2 Ontologies for cybersecurity

Another approach is to classify the kinds of attacks and security vulnerabilities into a traditional OWL-based Semantic Web ontology [24]. Although such work is interesting, and could be combined with provenance-based data integration in order to help, for example, the U.S. Department of Homeland Security in combining data about cyber-attacks, this approach does not actually discuss the problems inherent in using such ontologies within an adversarial environment.

2.3 Privacy ramifications of Open Data:

While publishing open data using Semantic Web standards may appear to be a public good, even innocuous public data such as road data may serve to de-anonymize users and so reveal personal or sensitive data, ranging from whether or not a property is abandoned to enabling the discovery of sexual preference [11]. In order to prevent these problems, the publication and search of encrypted RDF data has been proposed [21], as well as differential privacy [17], but so far neither have been implemented. Although there has been increased advocacy by Berners-Lee for the use of Semantic Web standards to "decentralize the Web" and provide personal data [22], the very same use of the Semantic Web for data integration can be used by third parties in order to build FBI Fusion Centers in the USA [31].

2.4 Semantic Web Security

Although previous research has noted on a very high-level "securing RDF is much more challenging" than traditional XML and HTML-based infrastructure as "we also need to ensure that security is preserved at the semantic level" [28], so far there is no research in this direction that go beyond the mere "idea of semantic web security standardization" [23] Instead, enterprise and government users simply believe that it is best to "access control or security occurs at the layer of the HTTP access and protocols, and not at the linked data layer." [3]

[3] http://structureddynamics.com/linked_data.html

3 Defining Semantic Security

What is security? Security is defined in terms of an attacker(or "threat model"). Informally, if a message is encrypted, an attacker can not discover the original message without a *secret key* that the attacker does not have. The original message is called the "cleartext" (M) and the message encrypted by the secret key is called the "ciphertext" (C). In terms of encryption and decryption, it is $E(M) \rightarrow C$ and $D(C) \rightarrow M$. To define this more precisely involves defining the property of *semantic security*, as defined by Goldwasser and Micali: "Whatever is efficiently computable about the cleartext given the ciphertext, is also efficiently computable without the ciphertext" [14], i.e. $P(f(M|C) = P(f(M))$ for any computable function f. To rephrase, an attacker can gain nothing in terms of information if the ciphertext has been intercepted. Formally semantic security is equivalent to indistinguishability under chosen plaintext attack (IND-CPA) by a computationally bounded adversary for a given bound ϵ. However, in this work for the sake of simplicity, definitions will given using the symbolic model, where the attacker is not considered computationally bounded and cryptographic primitives are considered functions over bitstrings in an abstract algebra [13]. This is a useful formalism as it can be easily automated by formal proof-proving tools [9]. The attacker breaks the security of a ciphertext if the attacker (A) knows the plaintext even if the message is encrypted, i.e. $knows(A, M) \wedge E(M)$.

Separable from security and less rigorously defined is *privacy*. Privacy is typically defined relative to an *anonymity set*, i.e. requiring the entity not being identifiable within a set of entities (the anonymity set) by an adversary. Privacy can be phrased in terms of *unlinkability*, namely that an entity can not be linked to their actions (in this context "link" means a information-theoretic reduction of the anonymity set, an entirely different notion than a hyperlink or link between RDF documents). On the Web, users are entities whose primary action is visiting a web-site, where a website is defined by an *origin*, where the origin is the domain name without the scheme, i.e. *origin.org* without the *http* scheme. The linking (i.e. "tracking") of users between origins violates their privacy. On the Web, the privacy and security boundary is given by the *same origin policy*: Any code or data on the user's browser is restricted by origin. So a cookie from *http://origin.org* should be accessible from *https://origin.org* and any sub-pages such as *http://origin.org/page.html*, but should not be accessible from *http://origin2.org*, and a user visiting the latter should not be connected by a third party (such as the website themselves are a third-party advertising bureau). The same goes for any state changes in the browser, including information in *localStorage* in the browser. T

4 The Lack of Network Layer Security (TLS) on the Semantic Web

4.1 Background

Transport Layer Security (TLS) is the well-known IETF standard for encrypting content transmitted over HTTP.[4] In addition to encrypting data sent over HTTP, the server that delivers the HTTP message can be *authenticated* with a certificate (a public key attached to its domain name), so that the origin can be proven to have sent the message.[5] Users do not authenticate to servers using TLS as specified by the IETF, but only the web site authenticates.

This means that resources hosted on URIs without TLS are vulnerable to having their content intercepted, which is equivalent to the adversary visiting the site rather than the user. Note the adversary can remove any additional messages (such as HTTP headers or certificates that are in plaintext) and replay the message without those messages. Worse, the content could be altered by an attacker without the possibility of a user knowing that this is the case, giving the user a fake resource.

Why not use TLS on the Semantic Web to preserve security and privacy? When a URI is enabled with TLS, the URI uses HTTPS as its scheme in the URI rather than HTTP. TLS was called informally "SSL" (Secure Sockets Layer), and HTTP with TLS enabled adds a "S" for historical reasons to become HTTPS. Unfortunately, the RDF specification states that HTTP and HTTPS URIs are not the same: "Two IRIs are equal if and only if they are equivalent under Simple String Comparison ... further normalization MUST NOT be performed when comparing IRIs for equality" [12]. Using *owl:sameAs* (or to preserve the correct semantics, *owl:equivalentClass* and *owl:equivalentProperty* when needed) between HTTP and HTTPS URIs does not work as these are statements about the things a URI denotes [16], not the text of the URI itself. RDF also does not have a way to talk about a URI itself via a mechanism such as quoting.

One could claim a Semantic Web URI is only a name and so the usage of TLS is not required. If there is no access to the content of the URI, URIs are the same as any other arbitrary string in a knowledge representation language, and so using HTTP or HTTPS has *no effect* on the formal semantics that determine the inferences that result from a Semantic Web reasoning engine. There would hold also be no problem if all Semantic Web resources were behind a perfectly secure enterprise firewall. Yet if this was truly the case then there really is no "Web" in the Semantic Web, and so the use of long HTTP URIs on the Semantic Web is simply an odd naming convention. The opposing view has also been argued by the Linked Data community that URIs should be linked to actual data that can be retrieved by Semantic Web applications [8]. The entire point of Linked

[4] TLS 1.2 is available at *https://tools.ietf.org/html/rfc5246*, with TLS 1.3 being under development at *https://tlswg.github.io/tls13-spec/*.

[5] In reality, the server proves it has a key validated by a Certificate Authority that the browser accepts. There is a long-standing issue that Certificate Authorities can create certificates for domains they do not own.[3]

Data is that a URI *should* have RDF statements available via HTTP *about* the resource(s) denoted by the URI. Further, if a user agent such as a browser can follow one URI to another via the links given by the RDF statements and so can "follow your nose" to discover new RDF statements that form a more complete knowledge representation of the resource.

4.2 Attacks

If TLS is not used on a origin, a network attacker may perform a number of attacks. The goal of the **network attacker** is to gain access to the plaintext of the resource retrieved from an origin, i.e. $knows(A, R)$. The first attack is trivial: If all data is sent over using HTTP, then the attacker can intercept the HTTP traffic. Also, an attacker can deliver *whatever data they want* to the unsuspecting user without detection due to the origin not being authenticated. For example, if one is retrieving open government data in Linked Data about the expenses given by the French government for the upkeep of the Eiffel Tower and the revenue created by tourists visiting the Eiffel Tower, and an attacker wanted to maliciously to prove to the French government that the Eiffel Tower was a bad investment, then the attacker could simply intercept the HTTP traffic hosted at the website and so change the numbers in the government data. It would be impossible for a user, such as a journalist, to tell if their HTTP traffic was tampered with by an attacker. This is a very easy attack that can be done using open-source tools such as *wireshark*[6] and *sslstrip*.[7]

Opportunistic encryption that automatically upgrades HTTP to HTTPS can be done by using the W3C Upgrade Insecure Requests specification where a server requests that a HTTPS URI be used if possible via a HTTP Header [32]. The server can then use a HSTS header to prevent any downgrade attacks that stripped the 'S' off the HTTPS in a URI [18]. In this way, a browser or Semantic Web application can ask for a Semantic Web HTTP URI and retrieve content from a HTTPS URI. The W3C began to use this methodology on its site, including RDF namespaces.[8] However, although the W3C has put hope in the eventual use of Upgrade Insecure Requests and HSTS, and to just "keep writing 'http:' and trust that the infrastructure will quietly switch over to TLS."[9] This path makes insecure HTTP URIs the default mode, and does not offer any reason for the Semantic Web to upgrade to TLS.

However, the attack is that the Upgrade Insecure Requests headers are delivered over HTTP, any attacker actively watching the unencrypted HTTP redirection can simply strip those headers to prevent upgrade to HTTPS and allow the HTTP content to be attacked by sending the same request, but stripped of the header. This allows the attacker to impersonate the user and intercept the

[6] *https://www.wireshark.org/*
[7] *https://moxie.org/software/sslstrip/*
[8] *https://www.w3.org/blog/2016/05/https-and-the-semantic-weblinked-data/*
[9] *https://www.w3.org/blog/2016/05/https-and-the-semantic-weblinked-data/#comment-93683*

unencrypted resource. So to just keep using HTTP URIs and hope for the best does not solve the problem.

4.3 Defenses

So why should Linked Data use HTTPS rather than HTTP URIs? URIs are also exposed to HTML links, and thus Tim Berners-Lee is rightfully worried that a switch to HTTP violates his principle that "Cool URIs Don't Change"[10] and so the adoption of HTTPS would break existing links. Berners-Lee goes even further:"Put simply, the HTTPS Everywhere campaign taken at face value completely breaks the web. In a way it is arguably a greater threat to the integrity for the web than anything else in its history"[4]. Indeed, network-level information about encryption (i.e. the use of HTTPS) should not have been exposed to the application level. One can still declare by fiat that all HTTPS URIs are equivalent to HTTP URIs on the Semantic Web: "If two URIs differ only in the 's' of 'https:', then they may never be used for different things."[11] However, this proposal has not been accepted and Upgrade Insecure Requests still requires an unencrypted HTTP response. The recommended solution should be to use HTTPS on all origins that host Linked Data. This is a serious problem for the Semantic Web community as the use of TLS-enabled HTTP URIs on the Semantic Web is minuscule, being less than *.1%* of Semantic Web URIs.[12] In comparsion in January 2017, the rest of the Web has *50%* usage of TLS.[13]

5 Application-level Attacks on the Semantic Web

Unfortunately, even if TLS is used correctly on the network level, there may be attacks on the level of the Web application by a Web attacker [2]. The goal of a **Web attacker** is to violate the privacy of a user as defined in Section 3. In terms of models, this means that not only can a user and an attacker visit websites to retrieve resources and gain knowledge of them, but now origins are actors that work on behalf of users and may thus interact with other origins (i.e. "visit sites") and have knowledge as well. This new paradigm can lead a Web attacker to have new attacks, in particular to impersonate a user to a given origin. As Semantic Web application-level standards do not use modern cryptography and violate the same origin policy, Web attackers can be successful. First, we'll look in detail at WebID+TLS in detail and then give an overview of security issues in a wider emerging stack of standards for distributed social networking from the W3C based on the Semantic Web.

---

[10] *https://www.w3.org/Provider/Style/URI.html*
[11] *https://lists.w3.org/Archives/Public/semantic-web/2014Aug/0078.html*
[12] According to *http://lodlaundromat.org/* (Retrieved on May 15th 2017).
[13] https://www.wired.com/2017/01/half-web-now-encrypted-makes-everyone-safer/

5.1 WebID+TLS considered Harmful

Background There has been some awareness of TLS in the Semantic Web community due to the *WebID+TLS* effort to use URIs as identifiers for people, with the evocative goal of creating decentralized social networking applications [26]. The goal of WebID is that each person can have their own personal URI that maintains their identity on the Web and from which their personal data, given by RDF statements, can be retrieved. Variations on WebID+TLS have tried adding access control in order make sure that personal data can only be given to explicitly authorized agents rather than given to absolutely anyone who issues a HTTP request [29]. In WebID+TLS, a user generates a client certificate using asymmetric cryptography (which contains a signature given by a private key stored in TLS keystore, as well as a link to their WebID URL) that is stored by the browser.[14] The public key can then be posted to the URI of their WebID URI. When a user wishes to authenticate themselves and authorize the transfer of their own personal RDF data from a site (called the *identity provider* on a distinct origin, following the terminology used in OAuth [15]) to a third-party (a *relying party*), a Semantic Web application can ask the user to authenticate a TLS session using their client certificate to identify their browser, and since the client certificate stored by the browser is signed with the private key that corresponds to the public key on their WebID URI (i.e. the signature on the certificate can be verified using that public key), they can prove that the WebID URI is controlled by the same agent that controls the browser. Normal TLS requires only the server authenticate, but client certificates also allow a client to authenticate, creating a mutually authenticated TLS session (not done in normal TLS). The Diagram 1 illustrates the protocol flow, with the flow of sensitive personal data given in green:

1. User presents a self-signed client certificate to the relying party.
2. Relying party extracts URI of identity provider from client certificate and retrieves public key from identity provider.
3. If public key matches key in client certificate, authenticate user using a challenge-response.
4. Relying party retrieves personal RDF data from he identity provider.

Attacks Despite using TLS, the problem with WebID+TLS is that it violates the security and privacy boundaries of the Web. In WebID+TLS, the client certificate is currently created with the `<keygen>` tag and stored in the TLS keystore. However, due to its age (it predates the same origin policy) and the lack of privacy concerns when it was designed, a *keygen*-generated client certificate can be accessible from any origin, and so serve as a "super-cookie" to track users across origins. In Proverif, for a hostile origin A where $(V(U, TLS(O), M_{cert}) \wedge V(U, TLS(A), M_{cert})) \rightarrow knows(A, K_1) \rightarrow knows(A, U)$. If an attacker wants to track a user, the attacker can query and ask for a client certificate. A malicious

[14] https://www.w3.org/2005/Incubator/webid/spec/tls/

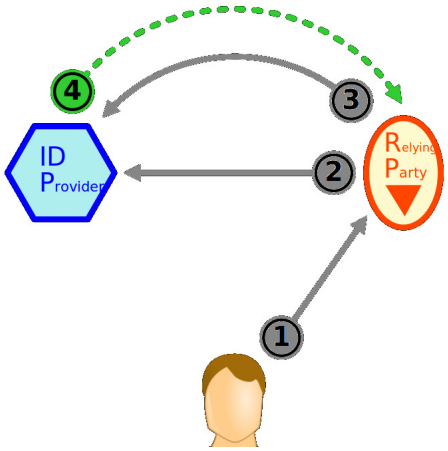

Fig. 1. WebID Protocol Flow

attacker from one origin who wanted to re-identify a user on another origin could simply install a client certificate on the malicious origin and ask for the certificate again on another origin. Worse, the current user-experience around both generating and selecting client certificates is confusing, and neither the installation nor usage of client certificates by HTML is standardized, so the usage of client certificates in HTML depends on ad-hoc browser behavior dependent on a particular idiosyncratic per-browser interpretation of a MIME-type. [15]

WebID+TLS is also based on out-of-date browser cryptography that is being deprecated by the browsers themselves. Even if the user does end up successfully identifying themselves with a client certificate, current browsers use the insecure MD5 hash function in the signed client certificate. MD5 has proven to not be collision resistant, which means that an attacker can generate a fake client certificate whose signature can be verified using the public key of a user even though the attacker does not have private key of the user [30]. In this way, a user can be impersonated by an attacker. This much more dangerous impersonation attack is $(V(U, TLS(O), M_{cert}) \wedge V(U, TLS(A), M_{cert})) \rightarrow knows(A, K_1) \rightarrow auth(A, O)$, with the last inference being possible as signatures can be faked due to broken cryptographic primitives.

Defenses Due to these security and privacy issues, browser vendors are currently deprecating *keygen* from HTML and client certificates handling from the application layer, which will mean *WebID+TLS* will stop working. Although the Semantic Web community has yet to engage with it, modern cryptographic primitives are now provided by the W3C Web Cryptography API.[16] Getting

[15] *https://groups.google.com/a/chromium.org/forum/#!msg/blink-dev/pX5NbX0Xack/kmHsyMGJZAMJ*

[16] https://www.w3.org/TR/WebCryptoAPI/

rid of passwords can be done via authentication by hardware tokens (or other authenticators) by the W3C Web Authentication API, which is designed both to not violate the same-origin policy (i.e. keys differ per origin) and use modern cryptographic primitives such as ECDSA [6]. Much like the rest of the Web, the Semantic Web can use explicit authorization of personal data transfer using IETF standards like OAuth [15]. By separating identities by origin and using modern cryptography, both the privacy and the security of users can be protected while enabling decentralized Semantic Web applications.

Part of the problem is a confusion over the layers of the Web: TLS is a network-level protocol, not an application level protocol. For example, interrupting a network-level TLS handshake in order to start a user-centric identity and authentication protocol in WebID+TLS is bad design insofar as it mixes the application-level concept of an "a person's identity" with the network level that just ships bits around. To some extent, the problems with the use of TLS on th e Semantic Web is that network level information (whether a HTTP connection is encrypted using TLS or not) is exposed on the level of a URI used in a Semantic Web applications, including but not limited to WebID+TLS. Web applications should use TLS to access origins but not use messages in the TLS flow such as certificates outside the network layer.

Instead, messages should be sent on the application level, as done in modern authorization protocols such as OAuth [15]. There has been some claims WebID+TLS is more private than OAuth. Although this is not true, as the identity provider is aware of all relying party transactions in both protocols, if user privacy against a malicious relying party is needed, then blind signatures can be used in OAuth to create unlinkability between a user's the identity provider and the relying party while still sending personal data [19].

5.2 Privacy and Security Flaws in W3C Social Web standards

Although a complete analysis of these standards would require more space, the precedent set by WebID+TLS is troubling insofar as other W3C Semantic Web standards and proposed standards do not take into account adversaries or modern cryptography as well. To continue our analysis of Solid, Solid allows a user after authentication via WebID+TLS to read and write data using HTTP GET and POST verbs with additional. However, again if a local POST is done to a HTTP origin rather than a HTTPS origin, all the attacks in Section 4 apply as the request and any Link headers will be unencrypted. Solid uses W3C Linked Data Notifications[17] to share RDF data using either the Linked Data Platform or a publish-subscribe model following the W3C WebSub Recommendation.[18]

While Linked Data Notifications requires signing data and a whitelist, but does not specify how either can be done. In terms of signatures, proposed standards such as Linked Data Signatures [19] repeat the mistakes of XML Digital

[17] https://www.w3.org/TR/ldn/
[18] https://www.w3.org/TR/websub/
[19] https://w3c-dvcg.github.io/ld-signatures/

Signatures: There is a complex canonicalization algorithm that does not specify how to resolve problems noted by earlier research [10] in chosing how to serialize a RDF graph in order (i.e. the "Expansion Algorithm" is undocumented in RDF Dataset normalization as used by Linked Data Signatures[20] and in the payload has a signature that can be simply detached by an attacker and replaced by the attacker's signature (a "signature stripping" attack) [1]. Linked Data Signatures ignores the foundations of digital signatures, namely that digital signatures require concrete byte-strings, and so do not work over abstract graphs without canonicalization. So it is unclear why Linked Data Signatures could simply convert the RDF to a binary string (via a "base64" transformation), the payload as is recommended by the IETF JOSE Working Group.[21]

WebSub claims to support digital signatures for authentication to prevent Sybil attacks (an attack where, due to lack of mutual authentication in publish-subscribe, server is overwhelmed by client requests or a server overwhelms a client with requests), but does not use asymmetric digital signatures, instead relying on a shared secret to create a HMAC (i.e. symmetric cryptography). Yet the shared secret is simply sent along optionally with the first message, and given a server must accept re-requests (possibly with new shared secrets), unless TLS connections are used on both the client and server in WebSub, the shared secret may be intercepted or re-set at any time. WebSub has only the callback URL of the subscriber require HTTPS, and the initial shared secret is sent back not with the callback URL but by the first subscription request. Lastly, the broken SHA-1 hash function is supported for calculation of the HMAC. These problems of authentication in scalable pub-sub systems are known to the research community, and solutions have been proposed relying on identity-based cryptography that are not used in WebSub [27]. Thus, it can be shown that the problems of authentication and the failure to use well-known cryptographic techniques continues to be a problem in further proposed Semantic Web standards.

6 Semantic Attackers

6.1 Background

While earlier attacks are generic results of either the lack of network-level encryption in TLS or bad protocol design, the Semantic Web enables a new class of attacker specific to RDF. In detail, this **Semantic attacker** has the goal of altering inference procedures. This kind of if the Semantic attacker builds on attacks on the network, and possibly the Web level, to compromise the RDF triples that an inference procedure depends on and so maliciously alter them, therefore changing results of the inference procedure. An attack is semantic insofar as it depends on the semantics of an inference procedure, although it may

[20] https://json-ld.github.io/normalization/spec/
[21] https://tools.ietf.org/html/rfc7515

but does not have to alter the semantics of the inference procedure itself.[22] Due to the distributed nature of resources on the Semantic Web.only gets worse if the Semantic Web application uses the "follow your nose" algorithm used in some Linked Data applications.

6.2 Attack

At *any point* data was retrieved from RDF statements given by a HTTP URI that does not use TLS, then the attacker can simply change the data and so influence the Semantic Web inference engine. An inference procedure I is given as depending on a set of x triples $R = R_1 \wedge R_2 \wedge ... R_x$ such that new triples S are produced by the inference procedure $(I(R) \rightarrow S)$. In Linked Data, these triples result from the visiting of a Semantic Web-enabled agent U visiting $(V \wedge R)$ websites O such that $V(U, O_1) \rightarrow R_1$. Therefore, $R^1 = V(U, O_1) \wedge V(U, O_1) \wedge ... R_x$. However, based on the network attack given in Section 4, unless $V(U, TLS(O_1))$, then an attacker can know the response $V(U, O_1)) \rightarrow knows(A, R_1)$. If an attacker knows the plaintext, they can alter it arbitrarily, such that plaintext $R_1 \rightarrow R_a$. Therefore, one or more malicious triples can be inserted $R^2 = R_1 \wedge R_a \wedge ... R_x$ and the inference procedure will produce a result that is at least partially under the control of the attacker, $I(R^2) \rightarrow S^2$ where $S^1 \neq S^2$.

For example, if a Linked Data-aware application was trying to determine if a person belonged to a particular class, such as the class of all terrorists. If a malicious semantic attacker noticed that one of the resources that the inference procedure depended on did not use TLS, as would be the case if the definition of a terrorist was hosted in a RDF Schema given by non-TLS website (such as the majority of non-W3C Semantic Web vocabularies). In this case, the semantic attacker would intercept and alter the definition of terrorist in the OWL file by removing a triple that stated that the crime must be classified as political by virtue of a certain list of government-approved definitions. In this way, a person who did a simple robbery could be classified as a terrorist by Semantic Web inference engine operating off of insecure resources in a FBI Fusion Center. If further triples were removed, all sorts of people could be mistakenly classified as a terrorists by a correctly operating inference engine using poor data.

6.3 Defenses

Given that Semantic Web reasoning in Linked Data depends on having trusted information, the *entire* process of reasoning and information retrieval must use TLS for *every* URI if the Semantic Web application is to be trusted. The only exception to this is that if the URI is not accessed, but merely used as an identifier. However, in this case the only trusted Semantic Web inference process is one that does not depend on the HTTP infrastructure. If any triples are

[22] Note that this is out of scope, as an adversary could simply add a triple to make a previously-valid OWL inference invalid, and so leaving only an RDF(S) inference possible.

derived from Web-level protocols, these protocols should also maintain their security properties and use TLS properly to prevent attacks.

7 Conclusion

Network and semantic attacks on the Semantic Web could be fixed if TLS was used on the Semantic Web. The only problem is outdated Semantic Web specifications. Shouldn't W3C Recommendations, rather than being considered as religious texts, be fixed to keep up with modern security? Although the RDF specification lack a way to discuss URIs themselves [12], it would make sense that best practices allow HTTP and HTTPS URIs to be equivalent. While this could be added to future editions of the specification, there is no reason to wait. If one believes that the use of URIs on the Semantic Web is still a marginal phenomena, the argument that this "breaks" existing Semantic Web software seems to assume that the Semantic Web is a mature technology with widespread adoption. The Semantic Web still is in its early stages, and so doing security right should take precedence over preserving broken software.

The future of HTTPS is also bright: TLS has improved and became easier to deploy as well, with certificates available for free and the protocol itself faster and more secure. For example, TLS version 1.3 [23] corrects a number of problems in TLS 1.2 such as an unclear state machine and attacks on TLS authentication [7]. These problems and more are fixed in TLS 1.3, and there is now provably secure implementations of TLS 1.3 that rely on fast elliptic curve cryptography. Earlier, the price of server-side certificates was considered too high and there was concerns over the hierarchical nature of the Certificate Authority system, as a Certificate Authority can issue a certificate that has global scope. These problems led to either Semantic Web researchers completely ignoring TLS due to supposed "security concerns" and for those Semantic technologies using TLS to such as WebID+TLS to want to use "self-signed" certificates instead of those from a certificate authority. Thanks to the effort by "Let's Encrypt," the price of server certificates is now free, so there is no reason not to use the high-security TLS certificates from "Let's Encrypt" other than the time investment to configure a Web server to use TLS.[24] Also, if one uses certificates and is worried about a rogue Certificate Authority issuing certificates incorrectly or being compromised by a malicious actor, the IETF Certificate Transparency standards, employed by Google, provides efficient auditing and search of certificates using a Merkle tree.[25]

Simply using the modern standards used by the rest of the Web such as OAuth [15] would defeat most Web attackers, allowing it to be taken more seriously as a real deployment platform for sensitive personal data.Given that the design of cryptographic protocols is difficult for even experts, it should be

[23] *https://github.com/tlswg/tls13-spec*
[24] *https://letsencrypt.org/*
[25] *https://www.certificate-transparency.org/*

no surprise that the creation of new protocols in W3C Semantic Web Working Groups that may not this expertise can lead to problems. Rather than use WebID+TLS, Semantic Web applications should respect the same-origin policy and use modern methods of authentication, such as the Web Authentication API, and authorization, such as OAuth, that respect the same origin policy. If cryptographic primitives are to be used on new protocols on the Semantic Web, they should use the primitives provided by modern APIs as provided by the W3C Web Cryptography API and not broken cryptographic primitives whose interaction with the browser are not normatively specified.

Given that TLS certificates are free and that the Semantic Web has still not reached widespread usage enough to argue that moving from HTTP to HTTPS would break real-world applications, Semantic Web tools and Semantic Web vocabularies should switch as soon as possible to using TLS-encrypted HTTPS URIs. There is no excuse not to use encryption if you want users to trust the Semantic Web. Both open data and personal data require security, and personal data in addition also requires privacy, and the building block for both is proper usage of TLS. Otherwise, the use of the Semantic Web will likely be replaced by technology, such as blockchains, that takes security on board in its design.

8 Acknowledgments

This work is funded in part by the European Commission H2020 European Commission through the NEXTLEAP Project (Grant No. 6882).

References

1.
2. Devdatta Akhawe, Adam Barth, Peifung E Lam, John Mitchell, and Dawn Song. Towards a formal foundation of web security. In *Computer Security Foundations Symposium (CSF), 2010 23rd IEEE*, pages 290–304. IEEE, 2010.
3. Hadi Asghari, Michel Van Eeten, Axel Arnbak, and Nico Van Eijk. Security economics in the HTTPS value chain. In *Twelfth Workshop on the Economics of Information Security (WEIS 2013), Washington, DC*, 2013.
4. Tim Berners-Lee. Web security - "HTTPS Everywhere" harmful, 2015. https://www.w3.org/DesignIssues/Security-NotTheS.html.
5. Tim Berners-Lee, Dan Connolly, Lalana Kagal, Yosi Scharf, and Jim Hendler. N3logic: A logical framework for the world wide web. *Theory and Practice of Logic Programming*, 8(03):249–269, 2008.
6. Vijay Bharadwaj, Hubert Le Van Gong, Dirk Balfanz, Alexei Czeskis, Arnar Birgisson, Jeff Hodges, Michael Jones, Rolf Lindemann, and J.C. Jones. Web Authentication: An API for accessing scoped credentials, 2016. https://www.w3.org/TR/webauthn/.
7. Karthikeyan Bhargavan, Antoine Delignat Lavaud, Cédric Fournet, Alfredo Pironti, and Pierre Yves Strub. Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS. In *2014 IEEE Symposium on Security and Privacy*, pages 98–113. IEEE, 2014.

8. Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, pages 205–227, 2009.
9. Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *Proceedings of the 14th IEEE Workshop on Computer Security Foundations*, CSFW '01, pages 82–, Washington, DC, USA, 2001. IEEE Computer Society.
10. Jeremy J Carroll. Signing rdf graphs. In *International Semantic Web Conference*, pages 369–384. Springer, 2003.
11. David Corsar, Peter Edwards, and John Nelson. Personal privacy and the web of linked data. In *Proceedings of the 2013th International Conference on Society, Privacy and the Semantic Web-Policy and Technology*, pages 11–21, 2013.
12. Richard Cyganiak, David Wood, and Markus Lanthaler. RDF 1.1 concepts and abstract syntax, 2014. https://www.w3.org/TR/rdf11-concepts/.
13. D. Dolev and A. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198 – 208, March 1983.
14. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.
15. Dickt Hardt. The Oauth 2.0 authorization framework, 2012. https://tools.ietf.org/html/rfc6749.
16. Patrick J Hayes and Harry Halpin. In defense of ambiguity. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 4(2):1–18, 2008.
17. Benjamin Heitmann, Felix Hermsen, and Stefan Decker. Towards the use of graph summaries for privacy enhancing release and querying of linked data.
18. J. Hodges, C. Jackson, and A. Barth. HTTP Strict Transport Security (hsts), 2012. https://tools.ietf.org/html/rfc6797.
19. Marios Isaakidis, Harry Halpin, and George Danezis. Unlimitid: Privacy-preserving federated identity management using algebraic macs. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, pages 139–142. ACM, 2016.
20. Lalana Kagal, Tim Finin, and Anupam Joshi. A policy based approach to security for the semantic web. In *International Semantic Web Conference*, pages 402–418. Springer, 2003.
21. Andreas Kasten, Ansgar Scherp, Frederik Armknecht, and Matthias Krause. Towards search on encrypted graph data. In *Proceedings of the Workshop on Society, Privacy and the Semantic Web-Policy and Technology*, pages 46–57, 2013.
22. Essam Mansour, Andrei Vlad Sambra, Sandro Hawke, Maged Zereba, Sarven Capadisli, Abdurrahman Ghanem, Ashraf Aboulnaga, and Tim Berners-Lee. A demonstration of the solid platform for social web applications. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 223–226. International World Wide Web Conferences Steering Committee, 2016.
23. Adis Medić and Adis Golubović. Making secure semantic web. *Universal Journal of Computer Science and Engineering Technology*, 1(2):99–104, 2010.
24. Alessandro Oltramari, Lorrie Faith Cranor, Robert J Walls, and Patrick Drew McDaniel. Building an ontology of cyber security. In *STIDS*, pages 54–61. Citeseer, 2014.
25. Sylvia Osborn. Mandatory access control and role-based access control revisited. In *Proceedings of the second ACM workshop on Role-based access control*, pages 31–40. ACM, 1997.
26. Henry Story, Bruno Harbulot, Ian Jacobi, and Mike Jones. FOAF+SSL: Restful authentication for the social web. In *Proceedings of the First Workshop on Trust and Privacy on the Social and Semantic Web (SPOT2009)*, 2009.

27. Muhammad Adnan Tariq, Boris Koldehofe, and Kurt Rothermel. Securing brokerless publish/subscribe systems using identity-based encryption. *IEEE transactions on parallel and distributed systems*, 25(2):518–528, 2014.
28. Bhavani Thuraisingham. Security standards for the semantic web. *Computer Standards & Interfaces*, 27(3):257–268, 2005.
29. Sebastian Tramp, Henry Story, Andrei Sambra, Philipp Frischmuth, Michael Martin, and Sören Auer. Extending the WebID protocol with access delegation. In *Proceedings of the Third International Conference on Consuming Linked Data-Volume 905*, pages 99–111. CEUR-WS. org, 2012.
30. Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 19–35. Springer, 2005.
31. K Krasnow Waterman and Samuel Wang. Prototyping fusion center information sharing; implementing policy reasoning over cross-jurisdictional data transactions occurring in a decentralized environment. In *Technologies for Homeland Security (HST), 2010 IEEE International Conference on*, pages 63–69. IEEE, 2010.
32. Mike West. Upgrade Insecure Requests, 2015. https://www.w3.org/TR/upgrade-insecure-requests/.