Volume 2, Issue 3 APR 2015

A STUDY OF CROSS SITE SCRIPTING COMPULSION AGAINST WEB APPLICATIONS

[1] K. Suma Rani, [2] D. Raman

[1]M.Tech Student, Department of CSE, Vardhaman college of engineering, Kacharam village,Shamshabad Mandal, Ranga Reddy District, Telangana, India.

[2] Associate Professor, Department of CSE, Vardhaman college of engineering, Kacharam village,Shamshabad Mandal, Ranga Reddy District, Telangana, India.

ABSTRACT— Due to their high sensible impact, Cross-Site Scripting (XSS) attacks have attracted plenty of attention from the safety community members. Within the same approach, less effective defense techniques are projected, addressing the causes and effects of XSS vulnerabilities. As a result, associate individual usually now not inject or perhaps execute absolute scripting code in many real-life eventualities. During this paper, we have a tendency to examine the attack surface that is still when XSS associated similar scripting attacks square measure purportedly quenched by preventing an assaulter from corporal punishment JavaScript code. we have a tendency to discuss the question of whether or not associate assault extremely desires JavaScript or similar practicality to do attacks aiming for data larceny. The stunning result's that associate assaulter can even abuse Cascading Style Sheets (CSS) together with alternative net techniques as plain hypertext markup language, inactive SVG pictures or font files. Through many case studies, we have a tendency to introduce the thus referred to as scriptless attacks associated demonstrate that an individual may not got to execute code to preserve his ability to extract sensitive data from well protected websites. a lot of exactly, we have a tendency to show that associate assaulter will use apparently benign options to make facet channel attacks that live and exfiltrate nearly absolute information displayed on a given web site. we have a tendency to conclude this paper with a discussion of potential mitigation techniques against this category of attacks. Additionally, we've got enforced a browser patch that allows an internet site to form an important determination on being loaded in an exceedingly detached read or pop-up window.

This approach proves helpful for bar of bound forms of attacks we have a tendency to here discuss.

***Index Terms*— Scriptless Attacks, XSS, CSS, SVG, HTML5, Attack Fonts.**

INTRODUCTION

In the era of web 2.0 technologies and cloud computing, a fashionable set of powerful on-line applications is offered at our disposal. These internet applications permit activities as on-line banking, starting business transactions at the net stores, composing e-mails that can contain sensitive info, or perhaps managing personal medical records on-line. It's thus solely natural to surprise what styles of measures square measure necessary to guard such knowledge, particularly in reference to security and privacy issues.

A distinguished real-life attack vector is Cross-Site Scripting (XSS), a kind of injection attack within which associate degree resister injects malicious scripts into associate degree otherwise benign (and trusted) web site [11]. Specifically, XSS provides associate degree assailant with associate degree possibility of manipulating an online page across totally different sites with the assistance of scripts. For this sort of attacks, JavaScript is usually used because the language of choice; once the malicious script executes, it's full access to all or any resources that belong to the trustworthy web site (e.g., cookies, authentication tokens, CSRF tokens). as a result of their high sensible impact, XSS attacks and connected browser-security analysis have attracted plenty of attention from the safety community throughout the recent years [20,22].

Going back one step, we have a tendency to note that XSS attacks ought to meet 3 preconditions guaranteeing their success:

1. **Injectability:** the assailant should be able to inject knowledge into the Document Object Model (DOM) rendered by the net browser.
2. **Executability:** if JavaScript (or the other code) is injected, it should be dead.
3. **Exfiltration Capability:** attacker-harvested knowledge should be delivered to a different domain or resource for additional analysis and exploitation.

The fact that XSS recently replaced SQL injection and connected server-side injection attacks because the beloved threat within the OWASP ranking indicates that these 3 preconditions square measure consummated by several internet applications. As ascertained higher as, many current mitigation approaches against XSS consider the second precondition, primarily since injectability is commonly a desired feature in several web 2.0 applications. net users square measure inspired to contribute content and knowledge exchange between totally different internet applications through the DOM is progressively used. Thus, server- and client-side XSS filters try and take away scripts from the injected content, or, they struggle to modify/replace these scripts in an exceedingly means that they're not dead within the browser's DOM. the standard advise reads: just in case we have a tendency to with success forestall injected JavaScript from being mirrored or dead, an online application may be thought-about secure against XSS attacks. Note that a browser's rendering engine is commonly utilized in different tools, like e-mail purchasers or instant messengers, to show hypertext markup language content. By default, scripting is disabled within these styles of code to forestall attacks like XSS in the context of e-mail process or instant electronic communication. Again, the defense approach is to temper the attacks by preventing the second precondition from occurring.

CONTRIBUTIONS.

In summary, we have a tendency to create the subsequent 3 contributions during this paper:

- We have a tendency to describe associate degree attack surface that's ensuing from a bound of scripting capabilities for untrusted content in fashionable internet applications. we have a tendency to show however associate degree assailant will deploy malicious code in an exceedingly heavily restricted execution context. we have a tendency to label this category scriptless attacks as a result of they are doing not ought to execute (JavaScript) code.
- we have a tendency to discuss many novel attack vectors that square measure subtle enough to extract sensitive knowledge (our running example pertains to getting mastercard numbers) from a given web site, doing therefore while not corporal punishment script code. The attacks utilize a sequence of benign options that combined along achieve associate degree attack vector inflicting knowledge leak. we have a tendency to demonstrate that proprietary options, moreover as W3C-standardized browser practicality, may be wont to concatenate harmless options to operate as a capable and powerful facet channel attack. The delineate attacks relate to Cross-Site Request Forgery (CSRF) and protection CSP and that

- they square measure appropriate for unseaworthy nearly impulsive knowledge displayed on a given web site. Moreover, we have a tendency to determine web- and SVG-fonts as powerful tools for aiding attackers in getting and exfiltrating sensitive knowledge from injected websites. We've got enforced proof-of-concept examples for all attacks.
- We have a tendency to elaborate on the prevailing defense mechanisms directed at scriptless attacks, specifically bearing on protection techniques like the Content Security Policy (CSP). Unluckily, we have a tendency to additionally determine gaps within the CSP-based protection and canopy the constraints of X-Frame-Options header with reference to scriptless attacks. Moreover, we have a tendency to introduce a brand new browser feature that we've got enforced for the Firefox browser in an exceedingly type of a patch useful for mitigating scriptless attacks. As an extra result, this feature can even assist the mitigation of many different attack techniques, like double-click-jacking and drag &amp; drop attacks [21].

ATTACK SURFACE AND SCENARIO

In the past few years, the bar for a fortunate attack has been considerably raised upon the introduction of the many afresh and complicated techniques preventing attacks against net applications. we tend to speculate that this can be principally caused by an outsized variety of revealed exploits, the increase of technologies connected to HTML5, and also the ever-growing quality of HTML usage in non-browser environments, i.e., a browser's ripping engine is employed in an exceedingly wide range of contexts like instant electronic communication tools like artificial language and Skype, e-mail purchasers like Outlook, spirit and Opera Mail, diversion hard- and software package, and ultimately operational systems like Windows eight. As a result, of these environments need protection from HTML-based attacks. This has junction rectifier to the steady development of various defense approaches (e.g., [6]). it's to boot price noting that the amount of users that install security extensions like NoScript is growing: NoScript blocks an outsized vary of attacks against web site users by merely prohibiting JavaScript execution. Consequently, attacks against net applications became tougher and a web site that deploys latest defense techniques will already resist an outsized variety of attack vectors.

Given of these defense ways, we tend to expect that attackers can thrive towards developing techniques that operate in rendering contexts that either don't permit script execution or heavily limit the capabilities of an dead script. for example, HTML5 suggests victimization sandboxed Iframes for untrusted content; these basically limit script execution up to completely interference it and that they can become crucial trust tokens for future net applications. a awfully basic question thence involves mind: will AN antagonist still perform malicious computations in such a restricted context?

The following list concisely describes some eventualities wherever HTML is employed in browsers or browser-like software package; however JavaScript is either restricted or utterly disabled for security and/or privacy reasons.

Our attack techniques target these eventualities as a result of scriptless attacks modify information outpouring even in such heavily restricted environments:

1. HTML5 Iframe sandbox: The HTML specification describes a feature that enables a web site to border discretionary information while not sanctioning it to execute scripts and similar active content. The thus referred to as Iframe sandbox is invoked by merely applying AN Iframe part with a sandbox attribute. By default, the sandbox is strict and blocks execution of any active content, type practicality, links targeting completely different views and plugin containers. The restrictions are relaxed by adding space-separated values to it attribute content. Thus, with these settings, developers will for example permit scripting however command access to parent frames, permit type practicality, or permit pop-ups and modal dialogs. Though sandboxed Iframes area unit presently solely offered in Google Chrome and Microsoft net individual, we tend to predict their wider adoption because the delineated feature seems within the HTML5 specification. A reduced version of sandboxed Iframes, tagged security restricted Iframes, has been offered within the terribly early versions of net individual, for instance in MSIE half dozen.0.

2. Content Security Policy (CSP): The Content Security Policy may be a projected and actively developed privacy and security tool. Specifically, it's offered in Mozilla Firefox and Google Chrome browsers. The CSP's purpose is that the protocol header and meta part primarily based restriction of content usage by the web site in question; a developer will for example direct the user agent to ignore in-line scripts, resources from across domains, event handlers, plugin information, and comparable resources like net fonts. In Section four we'll discuss however CSP in its current state will facilitate mitigating the attacks introduced in Section three.

3. NoScript and similar script-blockers: NoScript may be a rather in style Firefox extension composed and maintained. Apart from many options digressive for this work, NoScript's purpose is to dam untrusted script content on visited websites. Normally, all script and content sources aside from few trusty default origins area unit blocked. A selected user will decide whether or not to trust the content supply and modify it, either quickly or in an exceedingly permanent manner. NoScript was in scope of our research: we tend to tried to bypass its protection and gain a capability to execute malicious code despite its presence. allow us to underline that scriptless attacks have well-tried to be rather effective for this purpose.

4. Client-side XSS filters: many user agents give integrated XSS filters. this is applicable to Microsoft net individual and Google Chrome also as Firefox with the put in NoScript extension. Our scriptless attacks aim to bypass those filters and execute malicious code despite their presence. In many examples, we tend to were able to fulfill our objective, despite execution in reaction.

5. E-mail purchasers and instant messaging: As noted higher than, a browser's layout engine is typically not solely utilized by the browser itself, as many tools like e-mail purchasers and instant messengers equally use the offered HTML render the filter police work the attack and interference scripture engines for his or her functions. Mozilla spirit is mentioned as a selected example. By default, scripting is disabled during this form of software:

AN e-mail consumer permitting usage of JavaScript or maybe plugin content within the mail body might induce severe privacy implications. Scriptless attacks so offer a possible approach for attackers to execute malicious code regardless.

BEYOND SCRIPT-BASED ATTACKS

In this section, we have a tendency to discuss the technical details of the attacks we have a tendency to developed throughout our investigation of the attack surface associated with scriptless attacks. As we'll see, scriptless attacks will grant a possible answer to notwithstanding exfiltrate and steal sensitive info within the contexts delineate within the previous section, bypassing several of the out there defense solutions like sandboxed Iframes, script-blockers (i.e. NoScript), or client-side XSS filters. For the remainder of the paper, we have a tendency to assume Associate in nursing wrongdoer has the subsequent capabilities:

We illustrate our attacks with the assistance of a straightforward internet application that processes mastercard numbers – it are often compared to the Amazon internet store or similar websites applied with a back-end appropriate for process or delegation mastercard transactions. This internet application permits U.S.A. to demonstrate our attack vectors in a very proof-of-concept state of affairs. we have a tendency to specifically selected mastercard numbers' process for they incorporates solely sixteen digits like for instance 4000 1234 5678 9010. This permits U.S.A. to exfiltrate info in a very short quantity of your time. Note that our operations area unit applicable to different attack eventualities similarly and that we can for instance justify however one will steal CSRF tokens and different kinds of sensitive info with our methodology. moreover, we have a tendency to enforced a scriptless keylogger [18] that enables remote attackers to capture keystrokes entered on an internet page, even once JavaScript is disabled (this vulnerability is being half-tracked as CVE-2011-anonymized).

specifying those for a WOFF font, whimsical strings of

ATTACK PARTS

The attacks delineate within the following sections make the most of many customary browser options out there in fashionable user agents and outlined within the hypertext mark-up language and CSS3 specification drafts. we have a tendency to list and concisely justify these options before moving on to demonstrating however they'll be combined to comprise the operating attack vectors. additional specifically, we have a tendency to show however legitimate browser options are often abused to exfiltrate content or establish facet channels practical to get specific info from an internet browser. we have a tendency to found the subsequent browser options to be helpful building blocks in Constructing attacks:

1. Web-fonts supported SVG and WOFF: The hypertext mark-up language and CSS specifications suggest browser vendors to supply support for various web-font formats [23]. Among those area unit scalable Vector Graphics (SVG) fonts and internet Open Font Format (WOFF). Our attacks use these fonts and utilize their options to vary the properties of displayed web site content. SVG fonts permit Associate in Nursing wrongdoer to simply modify character and glyptography representations, modification look of single characters, and diversify their dimensions. it's attainable to easily use attributes like breadth to assure that sure characters don't have any dimensions by distribution "zero width", whereas different attributes might have distinct and attacker-controlled dimensions. WOFF together with CSS3 permits employing a feature referred to as discretionary ligatures or discourse alternatives. By virtually any length are often diagrammatical by one character (again given distinct dimensions for ultimate measuring purposes).

2. CSS-based Animations: With CSS primarily based animations, it's attainable to over time modification a good vary of CSS and DOM properties while not victimisation any script code [14]. The properties permitting modification via CSS animations area unit flagged by specification as animatable. Associate in nursing wrongdoer will use CSS animations to alter the breadth or height of a instrumentation close DOM nodes that hold sensitive info, to call one example. By having the ability to scale the instrumentation, the contained content are often forced to react in specific ways that to the dimension changes. One reaction would be to interrupt lines or overflow the instrumentation. Just in case those behaviors area unit measurable, animation will cause info leaks supported the temporal arrangement parameters of that specific behavior.

3. The CSS Content Property: CSS permits to use a property referred to as content to extract whimsical attribute values and show the worth either before, after, or rather than the chosen part [8]. The attribute price extraction are often triggered by the property price function's use attr. For a benign use-case of this feature, contemplate the subsequent situation: A developer desires to show the link uniform resource locator of all or elite links on her web site by merely rendering the content of the href attribute once displaying the link, however just for absolute link URLs. this can be possible by utilizing the subsequent CSS code:

```
a[href^=http://]:after{content:attr(href)}
```

This powerful feature also can be wont to extract sensitive attribute values like CSRF tokens, passwordfield-values and similar information. afterwards, they might be created visible outside the attribute context. Combining the extracted info with a font injection provides a strong

adversaries. The directive default-src enforces the user measuring lever and facet channel.

MITIGATION TECHNIQUES

In this section, we analyze existing attack mitigation techniques to determine what extent website owners and developers can protect against scriptless attacks. Acknowledging the wide range of possibilities for scriptless attacks (this publication only discusses two of potentially many more attacks' variations), we conclude that several layers of protection are necessary to effectively and holistically defend against CSS-, SVG- and HTML-based data leakage.

4.1 Content Security Policy (CSP)

CSP was originally developed by Mozilla and it is now specified as a draft by the W3C Web Application Security working group. The primary goal of CSP is to mitigate content injection vulnerabilities like cross-site scripting by determining at least one domain as a valid source for scripting code. To achieve this goal, one can use a directive like frame-src or sandbox. To provide an example, in the case of frame-src it is possible to let a supporting user agent check which frames can be embedded in a website. It is therefore possible to gain a fine granularity about the allowed content on a controllable website. Thus, CSP is capable of reducing the potential harmful effects of malicious code injection attacks. Note that CSP considers both arbitrary styles, inline CSS, and web fonts as possibly harmful and therefore provides matching rules.

In the context of our scriptless attacks, it would be desirable to restrict fundamental prerequisites to prevent a Web page (or rather the user) from being attacked. Therefore, we analyzed the given CSP directives with respect to the attacks we introduced in this paper. First, we have found that nearly all directives of the W3C draft, except for the directive report-uri for reporting policy violations, are helpful in preventing a website and its users from being affected by agent to execute – with one exception – the remaining directives of the draft with the given default source of the directive value. Before going into detail regarding the default-src influenced directives, it is important to know that pure injections with script or style sheet code into a vulnerable Web page cannot be detected by CSP. Thus, it is only possible to block the content of a file that is loaded from an external resource.

This leads to the ability of blocking malicious content that is included within an external file. A look at our attacks shows that it makes sense to use at least style-src and img-src of CSP to further reduce the attack surface. By specifying the style of the protected Web page with stylesrc, it is possible to restrict the access to undesirable CSS files. Therefore, CSS-based animations for reading DOM nodes or a usage of the CSS content property will no longer work in this case as an attacking tool. The same applies to img-src; as mentioned before, SVG files can be used to carry out scriptless attacks and intercept events, keystrokes and similar user interaction without using scripting technologies.

In consequence, blocking SVG files from another site and especially another domain is recommended for achieving a better level of security. Based on our example attacks, we also propose to use frame-src to restrict the resources of embedded frames as well as font-src for limiting external font sources.

Once the possibility of increasing the security by restricting external file resources has been made clear, we are left with a following consideration: can one restrict possible attack vectors inside the protected site? This is exactly the case when we use sandbox as a directive which is not controlled or set by default-src. It restricts the available content based on the HTML5 sandbox attribute values. This directive can therefore be used to for example deactivate the execution of scripts; hence, JavaScript-based

loaded in a detached view, one can mitigate several attack

attacks will not function. What was not considered to be dangerous is scriptless code. In our case, sandbox is just helpful if one is facing a typical scripting attack.

4.2 Detecting Detached Views

Several of the attacks we described in Section 3 can be leveraged by using Iframes and similar content framing techniques. Nevertheless, a website can easily deploy defensive measurements by simply using proper X-Frame-Options headers. Attackers, aware of that defense technique, have since started utilizing a different way and leverage pop-up windows and detached views to accomplish data leakage exploits and even clickjacking attacks without being affected by frame-busting code and X-Frame-Options headers.

Some of these attacks have been documented under the label double-clickjacking, while other techniques involve drag & drop operations of active content such as applets, or copy & paste operations into editable content areas across domains. Due to the extended attack surface, we want to stress that as far as modern browsers are concerned, there is no feasible way for a website to determine if it is being loaded in a detached view respective pop-up window or not.

In order to fix this problem, we created a patch for a recent version of the Web browser Firefox (Nightly 14.0a1, available as of April 2012), providing a possible solution to prevent the described attacks. The patch extends the well-known DOM window object by two additional properties: isPopup and loadedCrossDomain. Both properties are represented by a boolean value and can be accessed in a read-only manner by any website at any time. As the naming already suggests, window.isPopup is true only if the actual GUI window represented by the current DOM window object is a detached view. Likewise, window.loadedCrossDomain is true only if the current DOM window object was loaded cross-domain.

Allowing a website to determine whether it is being techniques at once. This includes several of the aforementioned scriptless attacks, double-clickjacking, drag & drop as well as several copy & paste attacks. We plan to discuss this patch with different browser development teams and evaluate how this technique can be adopted by several browsers to protect users against attacks.

Conclusion

In this paper, we tend to introduce a category of attacks against net applications and decision scriptless attacks. The key property of those attacks is that they are doing not deem the execution of JavaScript (or the other language) code. Instead, they're entirely supported normal browser options accessible in fashionable user agents and outlined within the current HTML and CSS3 specification drafts.

In a way, this type of attacks may be seen as a generalization of CSS-based history stealing [22] and similar attack vectors. We tend to mentioned many browser options helpful for scriptless attacks, covering a range of the way during which a person will access data or establish a facet channel. Moreover, we tend to conferred many scriptless attacks against an exemplary net application and incontestible however a person will with success get sensitive data like CSRF token or user-input by abusing legitimate browser ideas. Additionally, we tend to showed that AN person may also exfiltrate specific data and establish facet channels that create this attack possible. Whereas the attacks mentioned during this paper presumptively don't represent everything of the way to illegitimately retrieve sensitive user-data, we tend to believe that the attack parts mentioned by United States square measure of nice importance to alternative attack vectors. Therefore, an in depth analysis and any careful investigation relating attainable defense mechanisms can likely yield additional attack vectors. we tend to hope that this paper spurs analysis on attacks against net applications that aren't supported the execution of JavaScript code. As another contribution, we tend to introduce a browser patch that permits a web site to see if it's being loaded in a very detached read or pop-up window, showcasing mitigation technique for many sorts of attacks. At intervals our future work, we are going to examine additional ways in which for addressing and preventing scriptless attacks.

REFERENCES

[1] M. Balduzzi, C. Gimenez, D. Balzarotti, and E. Kirda.Automated Discovery of Parameter Pollution Vulnerabilities in Web Applications. In Network and Distributed System Security Symposium (NDSS), 2011.

[2] D. Baron. :visited support allows queries into global history. https://bugzilla.mozilla.org/147777, 2002.

[3] A. Barth, C. Jackson, and J. C. Mitchell. Robust Defenses for Cross-Site Request Forgery. In ACM Conference on Computer and Communications Security (CCS), 2008.

[4] D. Bates, A. Barth, and C. Jackson. Regular expressions considered harmful in client-side xss filters. In Proceedings of the 19th international conference on World wide web, pages 91–100. ACM, 2010.

[5] P. Bisht, T. Hinrichs, N. Skrupsky, R. Bobrowicz, and V. Venkatakrishnan. NoTamper: Automatic Blackbox Detection of Parameter Tampering Opportunities inWeb Applications. In ACM Conference on Computer and Communications Security (CCS), 2010.

[6] P. Bisht and V. Venkatakrishnan. XSS-GUARD: Precise Dynamic Prevention of Cross-Site Scripting Attacks. In Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA). Springer, 2008.

[7] A. Bortz and D. Boneh. Exposing Private Information by Timing Web Applications. In 16th International Conference on World Wide Web (WWW), 2007.

[8] B. Bos, T. ¸Celik, I. Hickson, and H. Wium Lie. Generated content, automatic numbering, and lists. http://www.w3.org/TR/CSS21/generate.html, June 2011.

[9] Z. Braniecki. CSS allows to check history via :visited. https://bugzilla.mozilla.org/224954, 2003.

[10] D. Brumley and D. Boneh. Remote Timing Attacks are Practical. In USENIX Security Symposium, 2003.

[11] CERT Coordination Center. Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests. http :// www . cert. org / advisories/CA-2000-02.html, 2000.

[12] S. Chen, R. Wang, X. Wang, and K. Zhang. Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow. In IEEE Symposium on Security and Privacy, 2010.

[13] C. Curtsinger, B. Livshits, B. Zorn, and C. Seifert. Zozzle: Fast and precise in-browser javascript malware detection. In USENIX Security Symposium, 2011.

[14] J. Daggett. CSS fonts module level 3. http://www.w3.org/TR/css3-fonts/, Oct. 2011.

[15] E. W. Felten and M. A. Schneider. Timing Attacks on Web Privacy. In ACM Conference on Computer and Communications Security (CCS), 2000.

[16] M. Heiderich. Content exfiltration using scrollbar detection and media queries. http://html5sec.org/scrollbar/test, June 2012.

[17] M. Heiderich. Measurement-based content exfiltration using smart scrollbars. http://html5sec.org/webkit/test, June 2012.

[18] M. Heiderich. Scriptless SVG Keylogger. http : // html5sec.org / keylogger, June 2012.

[19] M. Heiderich, T. Frosch, and T. Holz. IceShield: Detection and Mitigation of Malicious Websites with a Frozen DOM. In Recent Advances in Intrusion Detection (RAID), 2011.

[20] M. Heiderich, T. Frosch, M. Jensen, and T. Holz. Crouching Tiger – Hidden Payload: Security Risks of Scalable Vectors Graphics. In ACM Conference on Computer and Communications Security (CCS), 2011.

[21] D. Huang and C. Jackson. Clickjacking Attacks Unresolved. https://docs.google.com/document/\\pub?id=1hVcxPeCidZrM5acFH9ZoTYzg1D0VjkG3BDW_oUdn5qc,June 2011.

[22] C. Jackson, A. Bortz, D. Boneh, and J. C. Mitchell. Protecting Browser State From Web Privacy Attacks. In 15th International Conference on World Wide Web (WWW), 2006.

[23] D. Jackson, D. Hyatt, C. Marrin, S. Galineau, and L. D. Baron. CSS animations. http://dev.w3.org/csswg/css3-animations/, Mar.2012.

[24] A. Janc and L. Olejnik. Web Browser History Detection as a Real-World Privacy Threat. In European Symposium on Research in Computer Security (ESORICS), 2010.