The Need for a Coherent Web Security Policy Framework

Jeff Hodges and Andy Steingruebl
PayPal, Inc.
{Jeff.Hodges,asteingruebl}@PayPal.com

Abstract

Web-based malware and attacks are proliferating rapidly on the Internet. New web security mechanisms are also rapidly growing in number, although in an incoherent fashion. In this position paper, we give a brief overview of the ravaged web security landscape, and the various seemingly piece-wise approaches being taken to mitigate the threats. We then propose that with some cooperation, we can likely architect approaches that are more easily wielded and provide extensibility for the future. We provide thoughts on where and how to begin coordinating the work.

1. Introduction

Over the past few years, we have seen a proliferation of AJAX-based web applications (AJAX being shorthand for asynchronous JavaScript and XML), as well as Rich Internet Applications (RIAs), based on so-called Web 2.0 technologies. These applications bring both luscious eye-candy and convenient functionality—e.g. social networking—to their users, making them quite compelling. At the same time, we are seeing an increase in attacks against these applications and their underlying technologies [1]. The latter include (but aren't limited to) Cross-Site-Request Forgery (CSRF) -based attacks [2], content-sniffing cross-site-scripting (XSS) attacks [3], attacks against browsers supporting anti-XSS policies [4], clickjacking attacks [5], malvertising attacks [6], as well as man-in-the-middle (MITM) attacks against "secure" (e.g. Transport Layer Security (TLS/SSL)-based [7]) web sites along with distribution of the tools to carry out such attacks (e.g. sslstrip) [8].

During the same time period we have also witnessed the introduction of new web security indicators, techniques, and policy communication mechanisms sprinkled throughout the various layers of the Web and HTTP. We have a new cookie security flag called HTTPOnly [9]. We have the anti-clickjacking X-Frame-Options HTTP header [10], the Strict-Transport-Security HTTP header [11], anti-CSRF headers (e.g. Origin) [12], an anti-sniffing header (`X-Content-Type-Options: nosniff`) [13], various approaches to content restrictions [14] [15] and notably Mozilla's Content Security Policy (CSP; conveyed via a HTTP header) [16], the W3C's Cross-Origin Resource Sharing (CORS; also conveyed via a HTTP header) [17], as well as RIA security controls such as the `crossdomain.xml` file used to express a site's Adobe Flash security policy [18]. There's also the Application Boundaries Enforcer (ABE) [19], included as a part of NoScript [20], a popular Mozilla Firefox security extension. Sites can express their ABE rule-set at a well-known web address for downloading by individual clients [21], similarly to Flash's `crossdomain.xml`. Amidst this haphazard collage of new security mechanisms at least one browser vendor has even devised a new HTTP header that disables one of their newly created security features: witness the `X-XSS-Protection` header that disables the new anti-XSS features [22] in Microsoft's Internet Explorer 8 (IE8).

Additionally, there are various proposals aimed at addressing other facets of inherent web vulnerabilities, for example: JavaScript `postMessage`-based mashup communications [23], hypertext isolation techniques [24], and service security policies advertised via the Domain Name System (DNS) [25]. Going even further, there are efforts to redesign web browser architectures [26], of which Google Chrome and IE8 are deployed examples. An even more radical approach is exhibited in the Gazelle Web Browser [27], which features a browser kernel embodied in a multi-principal OS construction providing cross-principal protection and fair sharing of all system resources.

Not to be overlooked is the fact that even though there is a plethora of "standard" browser security features—e.g. the same origin policy, network-related restrictions, rules for third-party cookies, content-handling mechanisms, etc. [28]—they are not implemented uniformly in today's various popular browsers and RIA frameworks [29]. This makes life even harder for web site administrators in that allowances must be made in site security posture and

approaches in consideration of which browser a user may be wielding at any particular time.

Although industry and researchers collectively are aware of all the above issues, we observe that the responses to date have been issue-specific and uncoordinated. What we are ending up with looks perhaps similar to Frankenstein's monster [30]—a design with noble intents but whose final execution is an almost-random amalgamation of parts that do not work well together. It can even cause destruction on its own [31].

2. Towards a coherent site security policy framework

From our perspective as web site security practitioners, we believe that in the intermediate term it will be beneficial if we can work together with the goal of having deployed web browsers featuring more coherent security properties than they do today. We feel that cooperatively working to address specific subsets of the overall problem space will yield measurable results for both site operators and our users.

For example, we want to be able to deploy security policies for site-wide cookie handling, content restrictions, secure connection preferences, and various other things. We believe that continuing the current defacto practice of designing new, disjoint, HTTP headers for expressing individual facets of overall site security policies is not desirable for even the intermediate term. The individual headers, however expeditious in the near-term, should be replaced with a more generic security policy communication mechanism for the Web—a "website security policy framework". This policy communication mechanism must be secure and should have two facets, one for communicating securely out-of-band of the HTTP protocol to allow for secure client policy store bootstrapping, and then another in-band over HTTP/HTTPS for ease of policy delivery, configuration, and to leverage existing deployments.

For out-of-band secure client policy store bootstrapping, potential approaches are factory-installed web browser configuration, site security policy download a la Flash's `crossdomain.xml` and Maone's ABE for Web Authors [21], and DNS-based policy advertisement leveraging the security of DNS Security (DNSSEC) [32].

For in-band policy communication[1], we believe that a regime based on HTTP header(s) is appropriate. However we must devise a generalized, extensible HTTP security header(s) such that the on-going "bloat" of the number of disjoint HTTP security headers is mitigated and there is a documented framework that we can leverage as new approaches and/or threats emerge. It may be reasonable to devise a small set of headers to convey different classes of policies, e.g. web application content policies versus web application network capabilities policies.

In general, what we are striving for is to provide web site administrators the tools for managing, in a *least privilege* [33] manner, the overall security characteristics of their web site/applications when realized in the context of user agents.

Regardless of the overall approaches chosen for conveying site security policies, we believe that to be deployed at Internet-scale, and to be as widely usable as possible for both novice and expert alike, the overall solution approach will need to address these three points of tension:

1. *Granularity*: There has been much debate during the discussion of some policy mechanisms (e.g. CSP) as to how fine-grained such mechanisms should be. The argument against fine-grained mechanisms is that site administrators will cause themselves pain by instantiating policies that do not yield the intended results. E.g. simply copying the expressed policies of a similar site. The claim is that this would occur for various reasons stemming from the mechanisms' complexity [34].

2. *Configurability*: Not infrequently, the complexity of underlying facilities, e.g. in server software, is not well-packaged and thus administrators are obliged to learn more about the intricacies of these systems than otherwise might be necessary. This is sometimes used as an argument for "dumbing down" the capabilities of policy expression mechanisms [34].

3. *Usability*: Research shows that when security warnings are displayed, users are often given too much information as well as being allowed to relatively easily bypass the warnings and continue with their potentially compromising activity [35] [36] [37] [38] [39]. Thus users have become trained to "click through" security notifications "in order to get work done", though not infrequently rendering themselves insecure and perhaps compromised [40].

3. Discussion

As for the overall policy mechanism, we advocate a combination of CSP and ABE, or their employment in tandem, as a starting point for a multi-vendor approach. For a near-term policy delivery mechanism, we advocate use of both HTTP headers and a policy file at a well-known location. Leveraging DNSSEC is attractive in the intermediate term, i.e. as it becomes more widely deployed.

1The distinction between in-band and out-of-band signaling is difficult to characterize because some seemingly out-of-band mechanisms rely on the same protocols (HTTP/HTTPS) and infrastructure (transparent proxy servers) as the protocols they ostensibly protect.

In terms of granularity, vast arrays of stand-alone blog, wiki, hosted web account, and other "simple" web sites could ostensibly benefit from relatively simple, pre-determined policies. However, complex sites—e.g. payment, ecommerce, software-as-a-service, mashup sites, etc.—often differ in various ways, as well as being inherently complex implementation-wise. One-size-fits-all policies will generally not work well for them. Thus, we believe that to be effective for a broad array of web site and application types, the policy expression mechanism must fundamentally facilitate fine-grained control. For example, CSP offers such control. In order to address the less complex needs of the more simple classes of web sites, the policy expression mechanism could have a "macro"-like feature enabling "canned policy profiles". Or, the configuration facilities of various components of the web infrastructure can be enhanced to provide an appropriately simple veneer over the complexity.

Thus, with respect to configurability, development effort should be applied to creating easy-to-use administrative interfaces addressing the simple cases, like those mentioned above, while providing advanced administrators the tools to craft and manage fine-grained multi-faceted policies. Thus more casual or novice administrators can be aided in readily choosing, or be provided with, safe default policies while other classes of sites have the tools to craft the detailed policies they require. Examples of such an approach are Microsoft's "Packaging Wizard" [41] that easily auto-generates a quite complicated service deployment descriptor on behalf of less experienced administrators, and Firefox's simple Preferences dialog [42] as compared to its detailed `about:config` configuration editor page [43]. In both cases, simple usage by inexperienced users is anticipated and provided for on one hand, while complex tuning of the myriad underlying preferences is provided for on the other.

In the case of usability, much has been learned over the last few years about what does and does not work with respect to security indicators in web browsers and web pages, as noted above, these lessons should be applied to the security indicators rendered by new proposed security mechanisms. We believe that in cases of user agents venturing into insecure situations, their response should be to fail the connections by default without user recourse, rather than displaying warnings along with bypass mechanisms, as is current practice. For example, the Strict Transport Security specification stipulates the former hard-fail behavior.

4. Priorities

As described above, this is a multi-faceted problem space. We are not going to be able to attack all fronts at once. Though, a path forward does seem reasonably apparent. To us, the web policy mechanism and delivery work is the crucial piece to address first—portions of it are already reasonably well developed, e.g. CSP and ABE. However, coordination and cooperation will be essential going forward in order to end up with a coherent and extensible approach. Also, it should be a high priority for stakeholders to work to remove any perceived barriers to cooperative design, standardization, and wide implementation. Determining how backwards compatibility with the legacy inchoate approaches is addressed will be a key part of such an effort.

In terms of the implementations, as well as their configurability and usability aspects, explicit effort should be devoted to providing thorough support for less-experienced administrators and users. This means providing thorough configuration veneers/wizards, as well as likely performing further usability studies with respect to what might actually constitute a step forward in terms of security indicators and behaviors that will work for users in general. Egelman et al provide solid clues with respect to potential ways forward in this regard [35].

5. How and where to organize the effort?

Historically, the "browser market" has been characterized by vicious competition between browser vendors. It seems to outside observers that even security features have fallen prey to vendor oneupmanship, leaving both users and web site deployers in the lurch. Similarly, web server producers have had multiple battles over features, ease of configuration, and even versions of protocols supported.

Given the concerted attacks Internet outlaws are making on web-based ecommerce and users at large—thus blemishing the notion of online commerce in the eyes of many users (potential or current)—all involved in architecting and constructing the Web's underlying machinery should cooperate to move the ball forward. The relevant parties include but are not limited to web site deployers (e.g. PayPal), vendors of web servers, web browsers, RIA frameworks, application servers, and web application frameworks.

A particular difficulty in attacking the problem of security policy mechanisms for the Web is the lack of a single obviously appropriate forum. The two main standards bodies working in this space are the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C). Historically the IETF has worked on core protocols such as HTTP and the W3C has worked on the higher layers, e.g. HTML, XML, etc. Unfortunately many of the policies we envision fall between and/or overlap these two layers. They are

neither part of the core HTTP protocol (and relatives for our purposes such as TLS/SSL and DNS) nor are they properly part of HTML itself.

At this point we feel that the policy mechanism work having to do with network communications, e.g. STS and facets of ABE, as well as perhaps the policy delivery mechanism work, should occur in the IETF. We will be working towards that goal this year. Since CSP is specifically about content, the W3C is arguably a natural home for it, although its authors would have to say.

In any case we believe that the strengths of both of these standards bodies and perhaps others with particular skills in Human-Computer Interaction should be brought to bear on this problem space.

6. Conclusion

We believe the time is right for a concerted effort by various stakeholders to create a set of robust standards coherent Web security policy framework(s). We see the continued ad-hoc creation of new security mechanisms as inevitable, but with coordination and a well-specified applicable framework(s), we can maximize the benefit and reduce the risk of introducing such new security mechanisms. We believe that a generalized web security policy framework is within reach and is achievable in the near-to-intermediate-term.

7. Acknowledgements

We thank the anonymous reviewers, Ben Adida, and Bill Smith for their helpful suggestions and feedback.

8. References

[1] Breach Security, "THE WEB HACKING INCIDENTS DATABASE 2009," Aug. 2009. http://www.breach.com/resources/whitepapers/downloads/WP_TheWebHackingIncidents-2009.pdf

[2] R. Auger, *The Cross-Site Request Forgery (CSRF/XSRF) FAQ*, 2007. http://www.cgisecurity.com/articles/csrf-faq.shtml

[3] A. Barth, J. Caballero, and D. Song, "Secure Content Sniffing for Web Browsers--or How to Stop Papers from Reviewing Themselves," *Proceedings of the 30th IEEE Symposium on Security & Privacy*, Oakland, CA: 2009.

[4] D. Goodin, "Major IE8 flaw makes 'safe' sites unsafe - Microsoft's XSS buster busted," *The Register*, Nov. 2009. http://www.theregister.co.uk/2009/11/20/internet_explorer_security_flaw/

[5] J. Grossman, "Clickjacking: Web pages can see and hear you," Oct. 2008. http://jeremiahgrossman.blogspot.com/2008/10/clickjacking-web-pages-can-see-and-hear.html

[6] W. Salusky, *Malvertising*, 2007. http://isc.sans.org/diary.html?storyid=3727

[7] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC5246, Internet Engineering Task Force, Aug. 2008. http://www.ietf.org/rfc/rfc5246.txt

[8] M. Marlinspike, *SSLSTRIP*, 2009. http://www.thoughtcrime.org/software/sslstrip/

[9] *Scope of HTTPOnly Cookies*. http://docs.google.com/View?docid=dxxqgkd_0cvcqhsdw

[10] E. Lawrence, *IE8 Security Part VII: ClickJacking Defenses*, 2009. http://blogs.msdn.com/ie/archive/2009/01/27/ie8-security-part-vii-clickjacking-defenses.aspx

[11] J. Hodges, C. Jackson, and A. Barth, "Strict Transport Security," Dec. 2009. http://lists.w3.org/Archives/Public/www-archive/2009Dec/att-0048/draft-hodges-strict-transport-sec-06.plain.html

[12] A. Barth, C. Jackson, and I. Hickson, "The Web Origin Concept," Internet-Draft, work in progress, Internet Engineering Task Force, 2009. http://tools.ietf.org/html/draft-abarth-origin

[13] E. Lawrence, *IE8 Security Part VI: Beta 2 Update*, 2008. http://blogs.msdn.com/ie/archive/2008/09/02/ie8-security-part-vi-beta-2-update.aspx

[14] G. Markham, *Content restrictions*, 2007. http://www.gerv.net/security/content-restrictions/

[15] T. Jim, N. Swamy, and M. Hicks, "BEEP: Browser-Enforced Embedded Policies," *Proceedings of the 16th International World Wide Web Conference, Banff, Alberta, Canada, 2007*.

[16] B. Sterne and S. Stamm, "Content Security Policy (CSP)," 2009. https://wiki.mozilla.org/Security/CSP/Specification

[17] A.V. Kesteren, "Cross-Origin Resource Sharing (CORS)," Mar. 2009. http://www.w3.org/TR/2009/WD-cors-20090317/

[18] Adobe Systems, "Cross-domain policy file specification." http://learn.adobe.com/wiki/download/attachments/64389123/CrossDomain_PolicyFile_Specification.pdf?version=1

[19] G. Maone, *ABE - Application Boundaries Enforcer*, 2009. http://noscript.net/abe/

[20] G. Maone, *NoScript*. http://noscript.net/

[21] G. Maone, *ABE for Web Authors*, 2009. http://noscript.net/abe/web-authors.html

[22] Microsoft, “Event 1046 - Cross-Site Scripting Filter,” *MSDN Library*, undated.
http://msdn.microsoft.com/en-us/library/dd565647%28VS.85%29.aspx

[23] A. Barth, C. Jackson, and W. Li, “Attacks on JavaScript Mashup Communication,” *Proceedings of the Web 2.0 Security and Privacy Workshop,* 2009.

[24] M. Ter Louw, P. Bisht, and V. Venkatakrishnan, “Analysis of Hypertext Isolation Techniques for XSS Prevention,” *Proceedings of the Web 2.0 Security and Privacy Workshop*, 2008 .

[25] A. Ozment, S.E. Schechter, and R. Dhamija, “Web Sites Should Not Need to Rely on Users to Secure Communications,” *W3C Workshop on Transparency and Usability of Web Authentication*, 2006.

[26] C. Reis, A. Barth, and C. Pizano, “Browser Security: Lessons from Google Chrome,” *ACM Queue*, 2009, pp. 1-8.

[27] H.J. Wang, C. Grier, A. Moshchuk, S.T. King, P. Choudhury, and H. Venter, “The Multi-Principal OS Construction of the Gazelle Web Browser,” *USENIX Security Symposium*, 2009.

[28] M. Zalewski, *Browser Security Handbook*.
http://code.google.com/p/browsersec/

[29] A. Stamos, D. Thiel, and J. Osborne, *Living in the RIA World: Blurring the Line between Web and Desktop Security*, BlackHat presentation, iSecPartners, 2008.
https://www.isecpartners.com/files/RIA_World_BH_2008.pdf

[30] Mary Shelley, “*Frankenstein*, *or The Modern Prometheus,*” ca. 1831.
http://en.wikipedia.org/wiki/Frankenstein%27s_monster

[31] D. Goodin, “cPanel, Netgear and Linksys susceptible to nasty attack - Unholy Trinity,” *The Register*, 2009.
http://www.theregister.co.uk/2009/08/02/unholy_trinity_csrf/

[32] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, “DNS security introduction and requirements,” RFC4033, Internet Engineering Task Force, Mar. 2005.
http://www.ietf.org/rfc/rfc4033.txt

[33] J.H. Saltzer and M.D. Schroeder, “The Protection of Information in Computer Systems,” *Communications of the ACM*, vol. 17, Jul. 1974.

[34] I. Hickson and many others, “Comments on the Content Security Policy specification,” discussion on mozilla.dev.security newsgroup.
http://groups.google.com/group/mozilla.dev.security/browse_frm/thread/87ebe5cb9735d8ca?tvc=1&q=Comments+on+the+Content+Security+Policy+specification

[35] S. Egelman, L.F. Cranor, and J. Hong, “You've Been Warned: An Empirical Study of the Effectiveness of Web Browser Phishing Warnings,” *CHI 2008, April 5 - 10, 2008, Florence, Italy*, 2008.

[36] S.E. Schechter, R. Dhamija, A. Ozment, and I. Fischer, “The Emperor's New Security Indicators,” *Proceedings of the 2007 IEEE Symposium on Security and Privacy*.

[37] R. Dhamija and J.D. Tygar, “The Battle Against Phishing: Dynamic Security Skins,” *Proceedings of the 2005 Symposium on Usable Privacy and Security (SOUPS)*.

[38] J. Sobey, T. Whalen, R. Biddle, P.V. Oorschot, and A.S. Patrick, *Browser Interfaces and Extended Validation SSL Certificates: An Empirical Study*, Ottawa, Canada: School of Computer Science, Carleton University, 2009.

[39] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L.F. Cranor, “Crying Wolf: An Empirical Study of SSL Warning Effectiveness,” *USENIX Security Symposium*, 2009.

[40] C. Jackson and A. Barth, “ForceHTTPS: Protecting High-Security Web Sites from Network Attacks,” *Proceedings of the 17th International World Wide Web Conference (WWW)*, 2008.

[41] Microsoft, “Packaging Wizard.”
http://msdn.microsoft.com/en-us/library/aa157732(office.10).aspx

[42] Mozilla, “Options window.”
http://support.mozilla.com/en-US/kb/Options+window

[43] S. Yegulalp, “Hacking Firefox: The secrets of about:config,” *ComputerWorld*, May. 2007.
http://www.computerworld.com/s/article/9020880/Hacking_Firefox_The_secrets_of_about_config