SCC.442 Information System Penetration & Countermeasures

An Analysis of the OWASP Top Ten 2013

Ben Goldsworthy, 32098584
b.goldsworthy@lancaster.ac.uk — ben.goldsworthy@owasp.org

I. Introduction

The Open Web Application Security Project (OWASP) is an attempt to improve the general level of web application security across all industries via the production of educational materials, and other projects. One such project is the OWASP Top Ten, detailing what the Project considers to be the ten most pervasive web application security vulnerabilities at a given time. This report shall first summarise the OWASP itself, and then the Top Ten project and its methodolgy. The 2013 instalment shall be detailed, followed by a critical analysis of its choices. A number of suggestions shall then be made for a proposed 2017 update. Though this report shall be written from the perspective of an official OWASP Top Ten for 2017 not yet having been released, this is in fact not the case. For the genuine 2017 OWASP Top Ten, see OWASP 2017. Other than this mention, the 2017 Top Ten is considered to be outside of the scope of this report.

II. OWASP

The Open Web Application Security Project (OWASP) is worldwide not-for-profit organisation that aims to provide 'an open community dedicated to enabling organizations to develop, purchase, and maintain applications that can be trusted.' (OWASP 2013) Founded in 2001, it maintains a comprehensive wiki wherein one may find information about various sub-projects that fall under the OWASP umbrella. Offline, it exists worldwide in the form of volunteer-run Chapters, including Student Chapters at educational institutions. The project is stewarded by the charitable OWASP Foundation.

One such sub-project is the OWASP Top Ten, which the Project describes as 'a powerful awareness document for web application security [that] represents a broad consensus about the most critical security risks to web applications.' (OWASP 2006b) It is not, as it is sometimes misunderstood, a guide or process by which one should perform a security audit, nor a tool for penetration testing—it is simply a list of 'the Top 10 negative things not to do' (OWASP 2015, p. 21). This does not, however, mean it is of no use to a pentration tester. It can provide useful guide for initial vulnerability scans, and be vital in translating the impact of such vulnerabilities for the likely non-technical clients.

The methodology for assessing the risk posed by each entry into the Top Ten has changed over time, and is currently 'based on [the standard risk model] and is customized for application security.' (OWASP 2006c) This standard risk model takes the form of the formula

$$Risk = Likelihood \times Impact$$

Likelihood is currently determined via assessment of: the ease with which an attacker can exploit a given vulnerability; how difficult it is to detect such an intrusion; and the vulnerability's prevalence across a number of datasets provided by organisational members of the OWASP. This is tempered by an organisation's own assessment of the threat agents that it is likely to face.

Impact is determined by considering the technical impact of the exploit in terms of the CIA(A) triad—confidentiality, integrity, availability (and accountability)—and the business impact in terms of financial, reputational and privacy damage (OWASP 2013, p. 4). This business impact is again the responsibility of the businesses in question to self-determine.

The subjective nature of all of these elements means that there will, necessarily, be differing opinions as to the composition and ordering of the Top Ten. To ameliorate this, each Top Ten is based on a lengthy period of consultation with members of the OWASP community and application security industry professionals.

III. OWASP Top Ten 2013

The 2013 OWASP Top Ten is as follows:

A1 - Injection
A2 - Broken Authentication and Session Management
A3 - Cross-Site Scripting
A4 - Insecure Direct Object References
A5 - Security Misconfiguration
A6 - Sensitive Data Exposure
A7 - Missing Function-Level Access Control
A8 - Cross-Site Request Forgery
A9 - Using Components with Known Vulnerabilities
A10 - Unvalidated Redirects and Forwards

'The threat landscape for applications security constantly changes', and as such there were a number of differences made to the previous Top Ten—2010's—for this instalment. *A2 - Broken Authentication and Session Management* was moved up from *A3* due to increased prevalence, though the project maintainers did so with the caveat that it was 'probably because this area is being looked at harder, not because issues are actually more prevalent.' *A8 - Cross-Site Request Forgery*'s prevalence was found to have lowered, and so too its ranking. This acts potentially as a testament to the efficacy of the OWASP Top Ten itself, having featured the vulnerability consistently for the past six years. *A7 - Missing Function-Level Access Control* was a broadening of the previous *A8 - Failure to Restrict URL Access*, and *A6 - Sensitive Data Exposure* a merging of *A7 - Insecure Cryptographic Storage* and *A9 - Insufficient Transport Layer Protection*. Finally, *A9 - Using Components with Known Vulnerabilties* was spun out from *A6 - Security Misconfiguration*—itself moved up to *A5*—owing to the 'growth and depth of component based development' (ibid., p. 4).

IV. Critical Analysis

A1 - Injection undoubtedly deserves it's place at the top of the list. Despite having been first discovered as far back as 1998 (rain.forest.puppy 1998), a 2011 report claims to have still detected an average attack rate of '71 SQLi attempts an hour', with peaks of up to '800–1,300 times per hour.' (Imperva 2011) More contemporarily, the 2015 TalkTalk data breach—which cost the business upwards of £42m and led to the leaking of over 150,000 customer records—was found to have been the result of just such an injection attack (Culture, Media and Sport Committee 2016, s. 17). Injection certainly satisfies the OWASP criteria of prevalence, ease of exploitability and business impact, which more than make up for its ease of detection. The fact that PHP introduced its injection-resistant range of MySQLi functions back in 2004 further shows that this vulnerability is likely to remain a headache for the foreseeable future.

A2 - Broken Authentication and Session Management's place in the ranking is a stronger candidate for criticism. OWASP state, in declaring its prevalence to be 'widespread', that 'Developers frequently build custom authentication and session management schemes, but building these correctly is hard.' (OWASP 2013, p. 8) Whilst this may have been the case in 2013, the situation appears markedly different today. The rise in the use of web frameworks such as Ruby on Rails and Express since then, as identified by Vuksanovic and Sudarevic 2011; Okanovic 2014, has been explosive. Alongside this has been the rise of other web development technologies such as Node.js, as well as the 'growth and depth of component based development' that the Project itself has identified. These developments shall, presumably, have diminished the need for a developer 'roll their own' security suite. Assuming that this reduced prevalence is borne out by the data, the other levels of risk assessed for this vulnerability do not justify its continued second-place position in the Top Ten.

On the other side of the equation are the *A3 - Cross-Site Scripting* (XSS) vulnerabilities. They can claim a venerability comprable with that of injection attacks (Barrett 2010); despite this, they still account for up to half of all reported web exploits (WhiteHat Security 2016, p. 22). Strom 2017 expects such attacks 'to grow by 166 percent in 2017'. Large organisations have suffered XSS attacks in the past (Leyden 2008; Arthur 2010); they continue to suffer them now (Leyden 2017). Closer to home, even the Lancaster University

Students' Union website features a stored XSS vulnerability (despite its initial disclosure by this author almost a year and a half previously; see Goldsworthy 2016). The OWASP have incorrectly assessed the impact of such attacks to be only 'moderate'. Whilst an XSS attack may lead to limited damage to the compromised server, it can put every single client that visits that server at risk. This, in turn, can lead to a huge business impact depending on the regulatory climate in which a given business operates within. One prominent recent threat for which a XSS vulnerability can present an easy attack vector is exploit kits, which assess connecting clients' browsers and software to identify exploitable vulnerabilities (Palo Alto Networks 2016). Another less malicious, but still concerning, threat is that of in-browser cryptocurrency miners, or 'cryptojacking' (Newman 2017b). Additionally, the recently-discovered Meltdown and Spectre vulnerabilities are exploitable via Javascript running in the victim's browser (Lipp et al. 2018; Kocher et al. 2018). Though OWASP may class the detectability as 'easy', the sheer (and apparently growing) prevalence and phenomenal follow-on risks surely earns this vulnerability a higher position.

A4 - Insecure Direct Object References seems, too, to be inflated in its ranking, for largely the same reasons as *A2 - Broken Authentication and Session Management*—the increasing use of pre-designed components in the construction of web application would suggest an ever-shrinking attack surface of poorly-secured direct object references.

Next comes that perennial bugbear of the information security professional: *A5 - Security Misconfiguration*. As opposed to A2 and A4, this is a vulnerability that is potentially exacerbated by an overreliance on one-size-fits-all frameworks, in that a developer may find themselves in a state of complacence and fail to properly configure the settings of their framework to work for their specific application. Drawing on the (possibly apocryphal) story of the United States nuclear arsenal spending two decades protected by the access code of '00000000' (Blair 2004), however, one is inclined to conclude that security misconfiguration is an ineradicable vulnerability, too deeply ingrained within human nature and the complexity of modern security configuration to ever hope to remedy entirely. The best current measures for reducing its likelihood are automated scans that can detect misconfiguarations and out-of-date software, along with auto-patchers, though the impact of last year's WannaCry outbreak shows how little consideration is paid in some large organisations to such methods (Křoustek 2017). The OWASP Top Ten methodology currently does not afford a vulnerability any greater weighting commensurate with its ease of remediation; if it did so, *Security Misconfiguration* would no doubt rank substantially higher.

A6 - Sensitive Data Exposure is equally pervasive. The 2015 Ashley Madison affair is perhaps the most prominent recent example of this, with a joint Australian–Canadian report finding 'the level of security safeguards', which 'should have been commensurately high', were not. 'The investigation team found critical gaps in security coverage indicative of the absence of appropriate policies and practices.' (Office of the Australian Information Commissioner 2016). If Ashley Madison is to be considered as something of a 'cowboy outfit', then what of Uber (Wong 2017), or Yahoo! (Newman 2017a), or Equifax (Ivanovna 2017)? It would be foolhardy to underestimate the prevalance of sensitive, yet poorly- or un-secured, data currently being held by the world's companies and governments, let alone the impact of its exposure. Indeed, this impact is set to increase in future years, with the European Union's General Data Protection Regulation coming into force in May of this year.

A7 - Missing Function-Level Access Control seems poorly delineated from *A2 - Broken Authentication and Session Management*, and perhaps not strong enough of a unique threat to hold up as its own entry into the Top Ten.

A8 - Cross-Site Request Forgery (CSRF)'s positioning appears about accurate. Whilst it can have quite an impact, it requires a number of difficult steps on the part of the attacker, not least of all having to trick the victim into clicking on their malicious link. The more steps to the attack chain, the more chances there are to disrupt the attack. Additionally, the potential impact of a CSRF can be limited by paying attention to the other elements on the list—stringent access controls, for example, can limit the damage that any compromised user can inflict on the system.

A9 - Using Components with Known Vulnerabilities can easily be merged with *A5 - Security Misconfiguration*, and OWASP make little argument in favour of its separation beyond that 'the growth and depth of component

based development has significantly increased the risk of using known vulnerable components.' (OWASP 2013, p. 4) The securing and updating of these components seems to be well within the scope of this pre-existing entry.
Finally, *A10 - Unvalidated Redirects and Forwards* appears to be placed accurately for a vulnerability that, despite being easily-detectable and relatively uncommon, can still have a substantial impact.

There is a case to be made for an additional entry. As mentioned above, XSS attacks pose a far greater threat than their current ranking suggests. However, a distinction should be drawn between stored XSS attacks, and reflective ones. OWASP state that the latter require a victim be 'tricked into clicking on a malicious link, submitting a specially crafted form, or even just browsing to a malicious site' OWASP 2006a. This requires substantially more effort on the part of the attacker, whereas a stored XSS attack can be left alone for victims to come across; this hands-off approach can also make the perpatrator of a stored XSS attack more difficult to trace. As such, it appears necessary to separate *A3 - Cross-Site Scripting* into two separate sub-entries: *Stored Cross-Site Scripting* and *Reflective Cross-Site Scripting*.
Additionally, the lack of transport-layer security—the use of `http://` over `https://`—is here proposed as an entirely new addition (or, rather, a re-addition, having been previously been included as *A9 - Insufficient Transport Layer Protection* in the 2010 Top Ten). This issue may not persist for too much longer, if Mozilla's aggressive attempts to force the use of `https://` in their browsers have any success (Barnes 2015; Vyas and Dolanjski 2017; Kesteren 2018), but it seems worthwhile to better highlight the issue than to obfuscate it via merging with *A6 - Sensitive Data Exposure*, which would be better focused on just the storing of data on the business' end.

Thus, after the suggested re-rankings, consolidations, splittings and deletions, the Top Ten here proposed for 2017 is as follows:

Top Ten 2013	**Change**	**Proposed Top Ten 2017**
A1 - Injection	No change	A1 - Injection
A2 - Broken Authentication and Session Management	Broadened into A3	A2 - Stored Cross-Site Scripting
A3 - Cross-Site Scripting	Split into A2 and A7	A3 - Broken Access Control
A4 - Insecure Direct Object References	Merged into A3	A4 - Security Misconfiguration
A5 - Security Misconfiguration	Moved to A4	A5 - Storing Data Unencrypted
A6 - Sensitive Data Exposure	Amended to A5	A6 - Missing Access Control
A7 - Missing Function-Level Access Control	Amended to A6	A7 - Reflected Cross-Site Scripting
A8 - Cross-Site Request Forgery (CSRF)	No change	A8 - Cross-Site Request Forgery (CSRF)
A9 - Using Components with Known Vulnerabilities	Merged with A4	A9 - No Transport-Layer Security
A10 - Unvalidated Redirects and Forwards	No change	A10 - Unvalidated Redirects and Forwards

There is, however, one caveat to these suggestions. Ideally, *A1 - Injection* and *A2 - Stored Cross-Site Scripting* should be placed in joint first place. This would better drive home the point that each can be catastrophic for different stakeholders. Whilst an SQL injection attack can be ruinous for a business, deleting or leaking entire databases in the blink of an eye, XSS attacks can be equally devestating for the clients that visit the vulnerable sites. As developments in regulation increasingly lay the onus for such breaches at the business' doorstep, the implications of one's own web application being weaponised in this way could be just as destructive for the business itself.

V. Conclusion

The threat landscape has changed in many ways since the 2013 Top Ten was released. Unfortunately, some other threats persist even decades after they are first discovered. With both of these factors in mind, the 2013 Top Ten has been critiqued item-by-item along the same methodology used by the Project in its initial creation. A proposed 2017 update has been detailed, which features mergers, broadenings, additions and amendments. This update should, it is hoped, better capture the state of web application security at present. However, as previously stated, the OWASP Top Ten is inherently a subjective process, so this is by no means accompanied by a cast-iron guarantee of accuracy. Additionally, business-assessed impacts and threat profiles will necessarily change the weightings given to each element on a per-case basis, and this is entirely to be expected. If those businesses are looking at their operations in enough detail to be making these assessments, then the Top Ten will have fulfillled its purpose.

REFERENCES

Arthur, Charles (2010). *Twitter users including Sarah Brown hit by malicious hacker attack*. URL: https://www.theguardian.com/technology/blog/2010/sep/21/twitter-bug-malicious-exploit-xss.

Barnes, Richard (2015). *Deprecating Non-Secure HTTP*. URL: https://blog.mozilla.org/security/2015/04/30/deprecating-non-secure-http/.

Barrett, Michael (2010). *How Cross Site Scripting was discovered*. URL: http://www.thesecuritypractice.com/the_security_practice/2010/11/how-cross-site-scripting-was-discovered.html.

Blair, Bruce (2004). *Keeping Presidents in the Nuclear Dark (Episode #1: The Case of the Missing "Permissive Action Links")*. URL: https://web.archive.org/web/20040404013440/http://www.cdi.org/blair/permissive-action-links.cfm.

Culture, Media and Sport Committee (2016). *Cyber Security: Protection of Personal Data Online*.

Goldsworthy, Ben (2016). *'Found a problem with the site?': Everything that's wrong with the new LUSU website*. URL: https://bengoldsworthy.uk/2016/09/everything-thats-wrong-with-the-new-lusu-website/#javascript.

Imperva (2011). *Hacker Intelligence Summary Report — An Anatomy of a SQL Injection Attack*. URL: http://www.imperva.com/docs/HII_An_Anatomy_of_a_SQL_Injection_Attack_SQLi.pdf.

Ivanovna, Irina (2017). *Equifax ex-CEO: Hacked data wasn't encrypted*. URL: https://www.cbsnews.com/news/equifax-ex-ceo-hacked-data-wasnt-encrypted/.

Kesteren, Anne van (2018). *Secure Contexts Everywhere*. URL: https://blog.mozilla.org/security/2018/01/15/secure-contexts-everywhere/.

Kocher, Paul et al. (Jan. 2018). 'Spectre Attacks: Exploiting Speculative Execution'. In: *ArXiv e-prints*. arXiv: 1801.01203.

Křoustek, Jakub (2017). *WannaCry update: The worst ransomware outbreak in history*. URL: https://blog.avast.com/wannacry-update-the-worst-ransomware-outbreak-in-history.

Leyden, John (2008). *Facebook poked by XSS flaw*. URL: https://www.theregister.co.uk/2008/05/23/facebook_xss_flaw/.

– (2017). *XSS marks the spot: Steam vuln dangles potential phishing line*. URL: https://www.theregister.co.uk/2017/02/08/steam_vulnerability/.

Lipp, Moritz et al. (Jan. 2018). 'Meltdown'. In: *ArXiv e-prints*. arXiv: 1801.01207.

Newman, Lily (2017a). *Yahoo's 2013 Email Hack Actually Compromised Three Billion Accounts*. URL: https://www.wired.com/story/yahoo-breach-three-billion-accounts/.

– (2017b). *Your Browser Could Be Mining Cryptocurrency For a Stranger*. URL: https://www.wired.com/story/cryptojacking-cryptocurrency-mining-browser/.

Office of the Australian Information Commissioner (2016). *Joint investigation of Ashley Madison by the Privacy Commissioner of Canada and the Australian Privacy Commissioner and Acting Australian Information Commissioner*.

Okanovic, Vensada (2014). 'Web application development with component frameworks'. In: *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on*. IEEE, pp. 889–892.

OWASP (2006a). URL: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS).

– (2006b). *Category:OWASP Top Ten Project*. URL: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

– (2006c). *OWASP Risk Rating Methodology*. URL: https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology.

– (2013). *OWASP Top 10 - 2013: The Ten Most Critical Web Application Security Risks*. URL: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#tab=OWASP_Top_10_for_2013.

– (2015). *Application Security Verification Standard 3.0*. URL: https://www.owasp.org/images/3/33/OWASP_Application_Security_Verification_Standard_3.0.1.pdf.

– (2017). *OWASP Top 10 - 2017: The Ten Most Critical Web Application Security Risks*. URL: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf.

Palo Alto Networks (2016). *Exploit Kits: Getting in by Any Means Necessary*. URL: https : / / www . paloaltonetworks.com/resources/research/exploit-kits.

rain.forest.puppy (1998). 'NT Web Technology Vulnerabilities'. In: *Phrack* 8.54. URL: http://phrack.org/issues/54/8.html#article.

Strom, David (2017). *A Primer on Cross-Site Scripting (XSS)*. URL: https://securityintelligence.com/a-primer-on-cross-site-scripting-xss/.

Vuksanovic, Irena Petrijevcanin and Bojan Sudarevic (2011). 'Use of web application frameworks in the development of small applications'. In: *MIPRO, 2011 Proceedings of the 34th International Convention*. IEEE, pp. 458–462.

Vyas, Tanvi and Peter Dolanjski (2017). *Communicating the Dangers of Non-Secure HTTP*. URL: https://blog.mozilla.org/security/2017/01/20/communicating-the-dangers-of-non-secure-http/.

WhiteHat Security (2016). *Web Applications Security Statistics Report*. URL: https://www.whitehatsec.com/info/website-stats-report-2016-wp/.

Wong, Julia (2017). *Uber concealed massive hack that exposed data of 57m users and drivers*. URL: https://www.theguardian.com/technology/2017/nov/21/uber-data-hack-cyber-attack.