Session 2 Reading List

This is a list of useful documents that will enhance your understanding of the course material. The reading list is session-by-session. “Primary” material means something you would be expected to read if you are serious about the course, and may help you to do the weekly exercise; “Additional” material you may want to read if you would like to deepen your understanding on a specific area.

**Primary material**

- Graff and van Wyk: Secure Coding: Principles and Practices, chapter 4: Implementation, subsection “State’ on the Web”. Where you are at it, read the whole chapter 4, it isn’t that long.
- If you don’t know Cross-Site Scripting already, have a look at
  - [https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
  - [https://training.safecode.org/course/cross_site_scripting_xss_101](https://training.safecode.org/course/cross_site_scripting_xss_101)
- And similarly for CSRF:
  - [https://training.safecode.org/course/csrf_101](https://training.safecode.org/course/csrf_101)
- The sqlmap SQL injection discovery tool is a powerful thing. Have a look at its feature list at [https://github.com/sqlmapproject/sqlmap/wiki/Features](https://github.com/sqlmapproject/sqlmap/wiki/Features).
- Have a look at Django security documentation at [https://docs.djangoproject.com/en/dev/topics/security/](https://docs.djangoproject.com/en/dev/topics/security/). Of special note, read the section on XSS and think about how the “safe strings” concept ([https://docs.djangoproject.com/en/dev/ref/utils/#module-django.utils.safestring](https://docs.djangoproject.com/en/dev/ref/utils/#module-django.utils.safestring)) relates to tainted data and output encoding. Also read the section on CSRF and try to comprehend how the CSRF protection ([https://docs.djangoproject.com/en/dev/ref/contrib/csrf/#how-csrf-works](https://docs.djangoproject.com/en/dev/ref/contrib/csrf/#how-csrf-works)) works in Django. It doesn’t matter if you don’t have previous Python or Django exposure; the overall idea should be clear.
- For the course targets, the weekly exercise will get you the exposure to the attack proxies. The Burp Suite documentation is fairly good, so I recommend reading [http://portswigger.net/burp/help/suite_usingburp.html#vulns](http://portswigger.net/burp/help/suite_usingburp.html#vulns) and get a feel of how a web application security assessment would run (even if you would choose to use Zaproxy for the weekly exercise).

**Additional material**

- If you are truly interested in delving deeper in specifically web application security:
  - Stuttard and Pinto: *The Web Application Hacker’s Handbook, 2nd Edition*. This is a thick one, and definitely not needed for the course. But if you want to go forward in web application security testing, this one methodically gives you a
(long!) checklist of things you need to explore. Written by the author of the Burp Suite, it goes well with that specific tool. Unlike many “hacking” books out there, the book approaches web security testing as a methodical and sometimes boring exercise (which is what it is).

- **Zalewski:** *The Tangled Web.* This book is a seemingly neverending list of design and implementation failures in web browsers and technologies. It makes an entertaining reading if you have any background in web application implementation or standardisation, and opens your eyes to all the crud that there is in this space.

- A recent paper discusses the issues with hybrid JavaScript/native frameworks such as Cordova / PhoneGap. It describes the challenges involved in trying to enforce a same-origin security model in an environment where application rights are determined by capabilities. [http://www.cs.utexas.edu/~shmat/shmat_ndss14nofrak.pdf](http://www.cs.utexas.edu/~shmat/shmat_ndss14nofrak.pdf)

- Another recent paper is Bittau et al.: *Hacking Blind,* [http://www.scs.stanford.edu/~sorbo/brop/bittau-brop.pdf](http://www.scs.stanford.edu/~sorbo/brop/bittau-brop.pdf). This is a pretty cool example of performing binary exploitation over HTTP.