**AP Trust Security and Risk Management**

**Reference: ISO 16363: 5.2**

THIS IS A CHANGE TO SEE WHAT GITHUB DOES>

**The second document, "AP Trust Security and Risk Management / Reference: ISO 16363: 5.2," is out of date. The first paragraph talks about OWASP and DREAD. OWASP applies in particular to web-based applications, such as our admin interface (aka "Fluctus" or "the Rails app"). Generally, we want to ensure we've addressed the OWASP "top 10", which are** described **in** [the PDF here](http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf)**. DREAD is a more general means of assessing risks.**

**Here's how we address the OWASP top 10:**

1. **Injection - Our admin UI uses Ruby on Rails, which is designed not to allow SQL injection or command injection. It's widely used and well-vetted. LDAP injection doesn't apply to the APTrust system, because we're not using LDAP accounts or services.**
2. **Broken Authentication and Session Management - We're using a standard Rails authentication and session management. Again, this system is widely used and well vetted. The Rails community is pretty responsive about issuing patches for security problems, and we update our server packages every few weeks. (We can do that more often when we have a sys admin.)**
3. **Cross-site scripting does not apply to our Admin UI, because we do not allow users to enter any data directly. The system merely reports what's in our repository.**
4. **Insecure Object Direct**
5. **References. Our Admin UI does not use references to any objects. It displays read-only data fetched from Fedora. That data is transformed from XML to temporary Ruby data structures, so there are no direct references to objects that can be manipulated or changed.**
6. **Security Misconfiguration - We follow best practices in our security configuration, using restrictive file permissions, keeping unnecessary services off of the server, and keeping sensitive setting and connection information beyond our user's reach. We do not give descriptive error messages or reveal configuration data when errors occur on our public-facing servers. We could still stand to use a security review, to make sure the configuration is all in order.**
7. **Sensitive data exposure. Our Admin UI restricts all users to seeing only their own institution's data, and that's just metadata about what they've stored. The admin UI does not provide access to the actual contents of the archive. User passwords are encrypted, so if someone does steal them, they're useless. We don't have any other "traditional" sensitive data, like credit card numbers or SSNs.**
8. **Missing Function-Level Access Control - We're using a Ruby gem called Pundit, which forces you to explicitly define access control on every publicly-available function (such as listing intellectual objects, viewing details of a generic file, etc.) If you don't define access control for a function, Pundit will prevent all users from accessing that function. They'll just see an error message saying the access controls for that function are missing.**
9. **Cross-Site Request Forgery (CSRF) - We're using the Rails 4 CSRF token throughout the site. This is the standard way of preventing CSRF. There aren't many places a user can even enter data in the Admin UI other than where they enter their account preferences or change their password. In any case, they are protected.**
10. **Using Components with Known Vulnerabilities - We're only using standard Rails and gems that are actively maintained by the community. When gems are abandoned, we move to ones that are maintained. For example, we moved away from the old cancan authorization gem to the more commonly accepted Pundit gem last year. We update our system packages periodically, but we should do this more regularly.**
11. **Unvalidated Redirects and Forwards - This doesn't apply to us, because we do not redirect to any pages outside the APTrust site.**

**As for DREAD, the risks we want to avoid are 1) losing data and 2) inadvertently giving access to data that should be protected. There's a lot in place to protect against these risks. The big ones are 1) we don't expose the delete feature at all to users; only that APTrust admin has access to it, and 2) to get access to someone else's data you would have to compromise both their Fluctus login/password AND their AWS access keys. That's because, to get an item out of the repository, you have to restore it (which requires institution admin access to Fluctus) and then, after it's copied into their restoration bucket, you have to download it (which requires their AWS access key id and their AWS secret key).**

**Having two types of storage in two regions of the country mitigates against the risk of data loss from a regional disaster.**

**The biggest risk is losing all the data by not paying the AWS bill, and I believe we're mitigating against that with a fund that will pay the bill for a year of so if APTrust fails to pay.**

**Finally, the second paragraph in "AP Trust Security and Risk Management / Reference: ISO 16363: 5.2" needs to be updated. We have a new sys admin coming on next month, and his duties will include a review of system configuration and security.**