Return to Advance CAMP Wiki

Advance CAMP - Monday, Oct. 27, 2014, 3:30pm, White River Ballroom J

TOPIC: Non-Web SAML

CONVENER: Chris Phillips/Rob Carter

MAIN SCRIBE: Eric Goodman

ADDITIONAL CONTRIBUTORS: Jim Basney

# of ATTENDEES: 50

MAIN ISSUES DISCUSSED:

Chris is focused on general discussion of approaches and possible solutions.

Rob focused on sharing of PII data to a software-defined secure network pipe, but not in a web manner. Particularly concerned about "last mile" to the application. Referred to project moonshot.

> Is it a common case where the client that is being used is not sitting on a server that is directly controlled under their infrastructure?
>
> Yes, it does.
>
> Assertion: Security people will find the use of phishable credentials, whether mediated via SAML or otherwise, to be unacceptable. So if you approach the problem as "how do you support non-phishable credentials in a federated way?" the problem can be simplified.

Are Mobile Apps in scope of this discussion? Yes, but needs to be scoped to applications where presentation of information is appropriate via a browser and where it's not. (When presentation via web browser is appropriate, then "bypassing" the web browser to present a native login screen is going the "wrong way").

Different potential use cases:

- Securing web services/REST
- Securing IMAP/POP
- Securing SSH
    - See https://carmenwiki.osu.edu/display/CICIDM/Federating+SSH

Overall, unlikely to be "one solution that rules them all". Will call on a couple of people on various projects that are approaching these services.

Rhys spoke about Moonshot:

Does EAP, GSSAPI/SASL (Microsoft version), for support of federated non-web. Becoming productionalized later in the year. Support for eResearch people, and support will will likely be SSH kinds of services. Not clear that GSSAPI is the "one true solution" but it will work and is acceptable.

> Scott asks: Do people realize that a user might shell into one system, then type their federated credentials to SSH to a second system (meaning the credential isn't being entered at the entry point that would generally be preferred).
>
> Is there a difference between Scott's use case and logging in using a web browser running on my home machine?
>
> How does eduRoam get around this?
>
> > Uses EAP to tunnel the communication to the right place (and this is what Moonshot uses). However EAP does not provide any UI support: It doesn't tell the user that they are entering their credentials into a safe login screen.
>
> What if the client is trusted? If you presume that entering your credentials on a third party (client) application is secure, you're saying you trust the (third-party) client but not the server.

Looking at getting support into the native SSH on Debian.

Is Moonshot relevant to authenticating a webservice?

Rhys: "You can, but it may be a bit of overkill". They've been adding in elements to support machine to machine authentication. "Is it vanilla SPNEGO?" Yes, mostly. Moonshot uses Moonshot to bootstrap the trust in its trust framework.

Jim Basney: Did an example of a protected SSH with SAML ECP using GSSAPI against an off-the-shelf Shibboleth IdP. See https://github.com/jbasney/mech_saml_ec. Today using SAML

ECP to get certificates from CILogon (http://www.cilogon.org/ecp) for use with GSI-OpenSSH (http://grid.ncsa.illinois.edu/ssh/).

Scott talked about https://tools.ietf.org/html/draft-ietf-kitten-sasl-saml-ec a SAML profile definition for GSSAPI/SASL that can leverage the exisiting trust framework (metadata, endpoints)). There can be an issue authenticating endpoints of the GSSAPI conversation, because they don't bind in a way that you can leverage, e.g., TLS authentication.

How do you protect a webservice using SAML?

If the application initiating the conversation is on the user's client (e.g., a mobile app that makes rest calls directly), then that's one problem. If you're doing machine-to-machine communication, then don't use ECP, use certificates. We already have that problem pretty well solved?

Examples from the audience of use cases that need this:

- Canonical example of this is Office 365: Microsoft gets your credential and asserts it back to your institution.
    - Could solve this with Moonshot but need MS to implement it
- GIT is another example: expect to receive your credentials and then turn around and authenticating it to a (specific institution's) verifier (e.g., LDAP server).
- The use of web services directly to users (i.e., via a local client)
    - Within the institution, shouldn't you just have the client use a privileged account? I.e., authenticate the user at the client, then pull data on their behalf with a privileged account?
    - Doesn't work if you want the user to control the access rather than granting the access to the application. Similar to the LDAP problem of looking up data with the user's credential (except you want to pull out authN to a separate transaction) vs. a privileged account.
    - OAuth is probably the right solution (per Scott C), but the spec/implementation as it exists is not interoperable, and needs to be fixed.
        - Can be done, and people are doing it. Does that mean this isn't a problem?
        - Is this scalable?

What about mobile? Any thoughts or concerns? The concern about trusting apps that are (potentially framing a web browser to do the authentication) where the user still can't see where they are logging in.

CIC is considering suggesting Don't Frame the Browser so users can see where they are authenticating.

Are there non-SAML solutions? What about federating one-time passwords? Works for session based authentication, but not for things like IMAP.

What's the problem with using termporary token services (like OAuth) and time/IP/etc limitation? Mostly that users can't type them in.

Scott's opinion is that industry won't solve this problem, because they are happy with putting everything in the browser.

One solution is just say "these are passwords, they're basically broken, so don't worry about it beyond not entering it somewhere you don't trust". Use a CILogon-type solution? Where you use a third party to bootstrap token generation (via SAML, possibly ECP, possibly a raw browser), and then the app relies on the token from there forward. How expensive is it to extend the process beyond individual applications?

Rhys: If OpenSSH is the only problem we're trying solve, then we've massively overengineered the (Moonshot) solution!

May not be possible to provide a "general" solution, because the risk factors and limitations are so different across use cases.

Can some "one time" password solution be used as a general advice model? Some discussion of technical concerns of potential implementations.

Can this problem go away if you support 2FA? Perhaps, but may depend on the MFA implementation being used. If it doesn't map to the password delivery process currently used, then it may not work, because delivery method is a new requirement on all clients.

ACTIVITIES GOING FORWARD / NEXT STEPS:

Where should this conversation continue?

No clear point to have this discussion.
There is interested in the one-time password approach.

AI: Perhaps Scott and Chris will flesh this out more and re-present it, but still not clear where.
AI: Determine where to continue discussion.

If slides are used in the session, please ask presenters to convert their slides to PDF and email them to acamp-info@incommon.org