Security Engineering Fall 2015

Lecture 13 – (Web) Application Security

Fabio Massacci

Why Web Application Security is Important?

- ***Let's look at the process again***
  - Authentication
    - Client identifies itself (NB ITself, not HERself or HIMself)
    - System challenges client's authentication
    - Client responds and systems let it go
  - Authorization
    - Application decides if it has the rights to do stuff
    - Client does stuff it is authorized to do
    - If Clients tries to do unauthorized stuff, application blocks it
- ***WebApplication addresses the problem of***
  - "What if the "it" on the other side is not who s/he claims to be"?
  - What if the "it" on the other side does not send the right data?
    - "What if there is a bug in the application enforcing access?"

Why should you care?

- ***After all your case study is a UTM***
  - Lots of sensors and machines talking to each other, true but they are all controlled by a remote location and they are all specialized protocols and techniques
  - Why should we bother of web application security, XSS, SQLInjection, etc. etc.
- ***New Trends in Technologies***
  - Convergence of IT and OT Networks
  - HMI Interfaces
  - Engineering Workstations

Now and Then for Critical Infrastructures

- ***Good old times***
  - Operational Network is physically distinct from IT
  - Devices have very limited capabilities and used a very specialized language
  - Maintenance is performed by member of staff who used specialized machines owned by the company
- ***All this is expensive and difficult to manage***

- ***Now***
  - OT commands travels over IT network
  - Devices are general purpose with management Web Interface (eg Java)
  - Maintenance is performed by outsourced contractor who brings his laptop inside to diagnose/update stuff
- ***All this is cheap and easy to manage BUT***

UNIVERSITY OF TRENTO - Italy

The Shadow PC

- ***A true story***
  - A Energy Provider detect a severe malware attack on their IT network trying to get on the OT network
  - They block the network and do forensics
  - They finally track the source of the attack to one building in a remote site across the ocean in an Island
  - BUT … there is no computer in that building according their IT department…
  - They go and physically inspect it and there is no computer, it has power transmission machines but really no computer to control them…
- ***What happened?***

UNIVERSITY OF TRENTO - Italy

The Airport Shutdown

- ***The hacker – CNN March 10, 1997***
  - the unidentified hacker broke into a Bell Atlantic computer system, causing a crash that disabled the phone system at the airport for six hours.
  - The crash knocked out phone service at the control tower, airport security, the airport fire department, the weather service, and carriers that use the airport.
  - Also, the tower's main radio transmitter and another transmitter that activates runway lights were shut down, as well as a printer that controllers use to monitor flight progress.
- ***The contractor - Sep 26, 2014***
  - Somebody went in and shutdown everything that had to do with landing at this airport," said ABC News aviation and military consultant Steve Ganyard, a former Marine colonel.
  - "They shut down the lights, they shut down the instrument landing system. There was nobody to talk to."

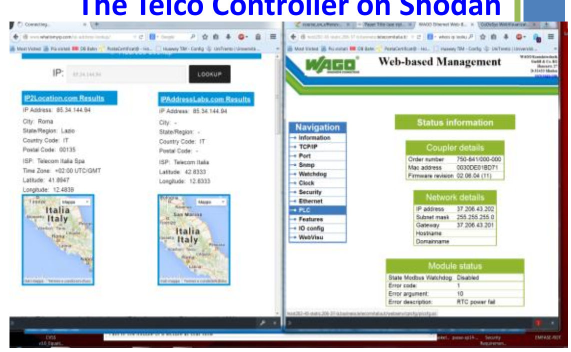
UNIVERSITY OF TRENTO - Italy

You don't believe it do you?

- ***"Personal Invulnerability" in Johnson's paper:***
  - Accidents only happen to incompetent people, or to systems or equipment designed by incompetent people
    - "accidents only happen when someone messes up, and I will not mess up, so no accidents will happen to me or the systems with which I work."
  - Because few engineers consider themselves to be incompetent, they are inclined to think that accidents will not happen to them or to the systems with which they are involved.
    - C. Johnson "Why System Safety Professionals Should Read Accident Reports"
- ***Let's search on the web for CODASYS***

UNIVERSITY OF TRENTO - Italy

The Telco Controller on Shodan



UNIVERSITY OF TRENTO - Italy

A bit of history explains a lot of things

- ***How and "why" the internet was invented?***
- ***TCP/IP + all services (DNS, etc.)***
  - Protocols to communicate among nodes of a trusted network (US Military + few Universities)
  - Essential to survive nuclear attacks → resilience is key
- ***HTTP + Web (Java 1.0, etc.)***
  - Protocol to communicate presentation of scientific data
  - Essential to be easy to use → usability is key
- ***Participants are all trusted***
  - They won't lie on who they are and
  - They won't send wrong data

UNIVERSITY OF TRENTO - Italy

Web Application Security Revisited – I

- ***At home***
  - Paolo: Daddy, I need 10€ to go to the movies
  - Fabio: Pick them from the wallet
  - Paolo: Where is the wallet?
  - Fabio: Near the entrance
  - Paolo: Thanks
- ***On the internet***
  - 192.37.15.6: Daddy, I need 10€ to go to the movies
  - Fabio: pick them from the e-wallet
  - 192.37.15.6: where is the e-wallet?
  - Fabio: on 193.37.18.67/server
  - 192.37.15.6: Thanks
  - Fabio: oh sh…

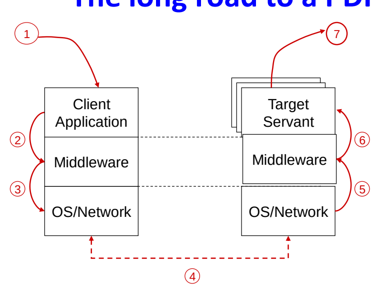
UNIVERSITY OF TRENTO - Italy

Web Application Security Revisited - II

- ***At home***
  - Luigi: Daddy, I need 500€
  - Fabio: What for?
  - Luigi: Going to buy A NEW LEGO MINDSTORM!!
  - Fabio: No need to shout, I can hear you. Anyhow, forget it.
- ***On the internet***
  - 192.37.15.6: Daddy, I need 500€
  - Fabio: What for?
  - 192.37.15.6: Going to buy QQQ***… 1GB of Qs…***QQQQQQQQ$%&//()=?éç°°è:;BATRFSIAa new Lego Mindstrom
  - Fabio: /dev/null… restarting on 193.37.18.67/server
  - 192.37.15.6: Thanks
  - Fabio: oh sh…

UNIVERSITY OF TRENTO - Italy

The long road to a PDP decision



1. User ask for access
2. App transfer info to Middelware
3. Middleware pass info to OS/Network
4. Info sent over the network
5. OS/pass info up to middleware
6. Middleware up to App
7. App finally makes a decision with what it got

Remote User AAA

- ***How can Server make decision on Client?***
- ***Identification with challenge-response***
  - Client sends identity
  - Server responds with random number
  - Client computes f(r,h(P)) and sends back
    - F and H are 1-way functions
    - P is the shared secret
  - Server compares value from user with own computed value, if match user authenticated
- ***Access control of application resources managed by server → conceptually just implementation***

HTTP Basic Authentication

- ***Client:***
  - GET /index.html HTTP/1.0
- ***Server:***
  - HTTP/1.1 401 Unauthorized
  - WWW-authenticate Basic realm="SecureArea"
- ***Client:***
  - GET /index.html HTTP/1.0}
  - Authorization: Basic
  - am9ldXNlcjphLmluQy5E
- ***Server:***
  - HTTP/1.1 200 Ok (plus document)
- ***Password sent in the clear, base64 encoded.***
- ***Not really secure:***
  - Step (4) → anybody who can see the user's reply learns the password.

HTTP Digest Authentication

- ***Challenge-response protocol (RFC 2617).***
- ***Server***
  - sends random challenge (nonce) to user.
- ***Client***
  - replies with hash (digest) of username+password+nonce+uri:
  - h(h(username:realm:password):nonce:h(method:digest-uri))
- ***Better security but still vulnerable to off-line dictionary attacks.***

Terminology: Nonces

- ***The term "nonce" was proposed Needham & Schroeder for unique values that are used only once.***
- ***A nonce can be a counter value, a time stamp, or a random number.***
- ***A nonce is not necessarily unpredictable.***
- ***Depending on the security goals, unpredictable nonces may be required.***

UNIVERSITY OF TRENTO - Italy

Off-line dictionary attacks revisited

- ***Use the password P to encrypt a randomly generated session key Ks; use session key to encrypt further data.***
  - A $\rightarrow$ B: encrypt$_P$(Ks)
  - B $\rightarrow$ A: encrypt$_{Ks}$(data)
- ***Vulnerable to off-line dictionary attack.***
  - Attacker guesses password P,
  - decrypts first message and gets a candidate session key K's
  - decrypt the second message with K's.
  - if result is meaningful text, → got P!

UNIVERSITY OF TRENTO - Italy

Encrypted Key Exchange (EKE)

- ***Step 0:***
  - user A generates a random public key/private key pair PubKa, PrivKa.
- ***Step 1:***
  - A sends public key pubKa to B, encrypted under the password P (symmetric encryption).
- ***Step 2:***
  - B randomly generates session key Ks;
  - sends Ks to A encrypted first under Ka (public-key enc.) and then under P (symmetric enc.)
- ***Protocol***
  - A $\rightarrow$ B: encryptP(PubKa)
  - B $\rightarrow$ A: encryptP(encryptPubKa(Ks))
  - A $\rightarrow$ B: encryptKs(data)

UNIVERSITY OF TRENTO - Italy

RADIUS

- ***RADIUS: Remote Authentication Dial-In User Service (RFC 2865).***
  - Centralized authentication, authorization, and accounting service.
  - Used for dial-up, virtual private network, wireless network access.
- ***RADIUS client and RADIUS server have***
  - common shared secret (password).
- ***Access-Request:***
  - user name, user password, authenticator, ID of client, Port ID which the user is accessing.

UNIVERSITY OF TRENTO - Italy

RADIUS (2)

- ***RADIUS server validates the sending client.***
- ***The server has a user database***
  - a user entry in the database lists the requirements which must be met to allow access.
  - A request from a client for which the server does not have a shared secret MUST be silently discarded.
- ***Always includes verification of password, can also specify client(s) or port(s) to which the user is allowed access.***
- ***Challenge-response authentication optional.***

UNIVERSITY OF TRENTO - Italy

eit Digital MASTER SCHOOL

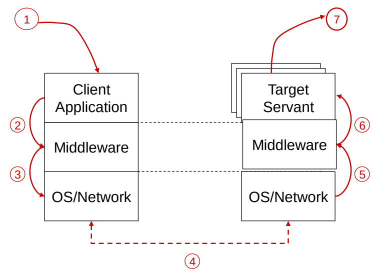
RADIUS (3)

- ***When password is present in the request, it is hidden using a method based on a has function (was MD5).***
- ***Passwords divided in blocks p1, p2, ..., pn.***
- ***Ciphertext blocks c1, c2,..., cn.***
- ***Secret S, random authenticator RA:***
  - c1 = p1 ⊕ MD5(S || RA)
    c2 = p2 ⊕ MD5(S || c1)
    .
    .
    .
    cn = pn ⊕ MD5(S || cn-1)
- ***Without challenge response still vulnerable to dictionary attacks but more difficult***

UNIVERSITY OF TRENTO - Italy

eit Digital MASTER SCHOOL

Only secure a part of the picture



1. User ask for access
2. App transfer info to Middelware
3. Middleware pass info to OS/Network
4. Info sent over the network
5. OS/pass info up to middleware
6. Middleware up to App
7. App finally makes a decision with what it got

UNIVERSITY OF TRENTO - Italy

eit Digital MASTER SCHOOL

Basic Problems still hanging

- ***At network level not yet very secure***
  - The OS may be subverted
  - The Network may be spoofed/manipulated etc.
  - More details in the last part of the course on infrastructure (Network/OS) security
- ***We need to secure the application but***
  - The applications may have bugs → forthcoming OWASP top 10 lecture
  - The application may be a controller… it is not a server → may have no clue who you are

UNIVERSITY OF TRENTO - Italy

eit Digital MASTER SCHOOL

It Takes 3 to Tango a AAA

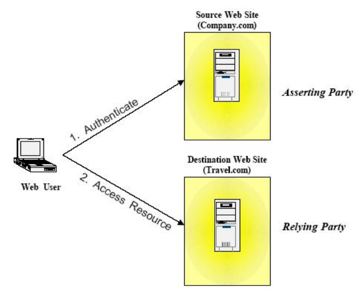
- ***The Asserting Party***
  - Who asserts information about a subject (has authority to grant/deny access to a user)
  - Asserts that a user has been authenticated and has been given associated attributes.
    - E.g.: This user is John Doe, has the email address john.doe@acompany.com, and was authenticated into this system using a password mechanism.
- ***The Relying Party***
  - Who wants to grant/deny access to a user on information supplied to it by the asserting party.
  - It is up to the relying party as to whether it trusts the assertions provided to it.
- ***The Client***

SAML Overview

- *SAML*
  - User authentication in distributed system uses Web Services.
  - SAML requirements driven by use cases.
- *Main use case: Web Single Sign-On (SSO).*
  - Allows users to gain access to website resources in multiple domains without having to re-authenticate after initially logging in to the first domain.
  - The domains need to form a trust relationship before they can share an understanding of the user's identity
- *New incarnation*
  - OpenAuth protocol → same concept with OpenData buzzword

25

Web Single Sign-On (SSO)



26

Travel Bookings

- *Scenario*
  - Authenticated users of Company.com need to access protected resources at Travel.com in order to make travel arrangements.
- *Company.com users*
  - should not need to have to re-authenticate to Travel.com
  - Only certain privileged users may book international travel
- *SSO scenario (without control on user) is just the "login with Gmail button" scenario*

27

Goods Purchasing

- *Authenticated users of Company.com use an internal purchasing system to place orders for office supplies from Supplier.com.*
- *Supplier.com needs to know*
  - user data → name and shipping address.
  - User authorization → whether user is authorized to purchase goods of that value or larger

28

UNIVERSITY OF TRENTO - Italy

Alternative: Browser cookies

- ***In the past,***
  - most SSO products used browser cookies to maintain state so that re-authentication is not required.
- ***However,***
  - browser cookies are not transferred between DNS domains.
- ***So,***
  - a cookie from www.abc.com will not be sent in any HTTP messages to www.xyz.com.
  - This could even apply within an organization that has separate DNS domains.

29

UNIVERSITY OF TRENTO - Italy

Centralized UTM Control Center

- ***Central portal system maintaining the authentication information for all users, linked to a number of satellite systems.***
- ***Satellite systems use access management products from a variety of vendors.***
- ***Users should only be required to be authenticated once, and can either go initially to the satellite system or the central portal.***
- ***The portal is the asserting party for the whole system, the satellite systems are the relying parties.***

30

UNIVERSITY OF TRENTO - Italy

SSO interoperability

- ***With proprietary cross-domain SSO products, organizations that want to perform cross-domain SSO have to use the same SSO product in all the domains.***
- ***This holds for SSO within one organization and for SSO across trading partners.***
- ***A solution based on web services can address this interoperability issue.***

31

UNIVERSITY OF TRENTO - Italy

SAML Concepts

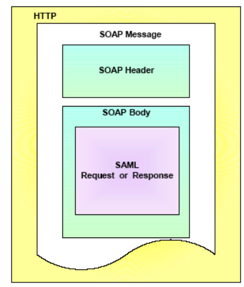
- ***Assertion: A package of information that supplies one or more statements made by a SAML authority.***
  - *Authentication statements* say "This subject was authenticated by this means at this time."
  - *Attribute statements* provide specific details about the subject (e.g., a user holds "Gold" status).
  - *Authorization decision statements* say what the subject is entitled to do.
- ***Protocol: SAML defines a request/response protocol for obtaining assertions.***

32

SAML Concepts

- *Bindings*: ***Detail how the SAML protocol maps onto transport and messaging protocols.***
  - SAML-SOAP binding (SAML over SOAP over HTTP).
  - Reverse SOAP (PAOS) binding.
  - HTTP post binding
  - SAML URI binding
- *Profiles*: ***Technical descriptions of particular flows of assertions and protocol messages that define how to use SAML for a particular purpose; derived from use cases.***

SOAP over HTTP binding



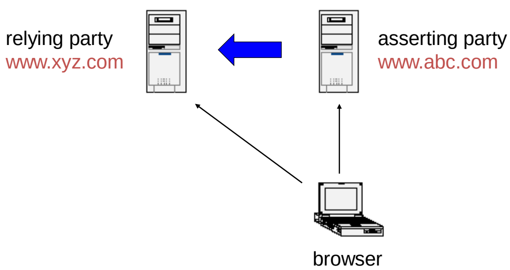
SAML Profiles

- *Browser/Artifact Profile*: ***Pull model***
- *Browser/POST Profile*: ***Push model: assertions POSTed (using the HTTP POST command) directly to the relying party.***
- ***Profiles assume:***
  - Use of a standard commercial web browser using either HTTP or HTTPS.
  - The user has been authenticated at the local source site.
  - The assertion's subject refers implicitly to the user that has been authenticated.

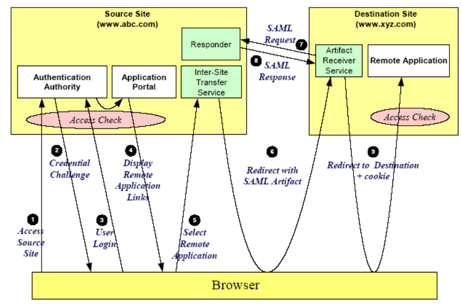
Browser/Artifact Profile

- ***Scenario***
  - A user has an authenticated session on the local source site and wants to access a resource on the destination web site and is directed there.
- ***In the HTTP message, an HTTP query variable is passed called an*** *artifact*:
  - a base-64 encoded string consisting of a unique identity of the source site (Source ID) and a unique reference to the assertion (AssertionHandle).
- ***The destination site (relying party) sends a SAML request containing the artifact to the local site (asserting party).***
- ***The assertions about the user are transferred back in a SAML response.***

Browser/Artifact Profile



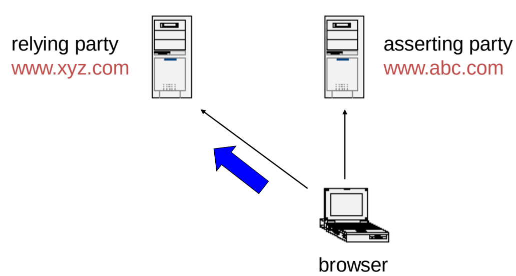
Detailed Processing for the Source-Site-First Scenario



Browser/POST Profile

- ***Scenario***
  - A user has an authenticated session on the local source site (asserting party) and wants to access a resource on the destination web site (relying party).
- ***An HTML form with the assertion about the user is provided back to the browser from the source site.***
  - The form contains a button (or other type of trigger, or JavaScript "auto-submit" action) that causes a POST of the assertion to the destination site to occur.
- ***The destination site makes its decisions based on the assertions contained within the POST message.***

Browser/POST Profile



Summary

- ***SAML addresses an aspect of access control in distributed applications:***
  - the entity managing the resource need not know about the subject requesting access.
- ***SAML defines message flows, but not protocols.***
  - We need protocols whereby an entity that can authenticate the subject transmits this information to the entity managing the resource.
- ***How does the relying party trust what is being asserted?***
  - How do prevent man-in-the-middle attacks?
  - The primary security mechanism is for the relying and asserting party to have a pre-existing trust relationship, typically involving a Public Key Infrastructure (PKI).

Security Analysis

- ***We need to add a bit of crypto for message and origin authntication***
  - Where message integrity and message confidentiality are required,
    - HTTP over SSL 3.0 or TLS 1.0 is recommended.
  - When an assertion is requested from an asserting party,
    - bi-lateral authentication is required
    - SSL 3.0 or TLS 1.0 using server and client authentication are recommended.
  - When an assertion is pushed to a relying party,
    - the response message be digitally signed using the XML digital signature standardì
- ***TLS/SSL we will see them in Infrastructure/Network Security Part***