Final exam in

Web Security EITF05

Department of Electrical and Information Technology
Lund University

October 22nd, 2013, 8.00-13.00

- You may answer in either Swedish or English.
- If any data is lacking, make (and state) reasonable assumptions.
- Use legible hand writing. If your answers cannot be read, you will receive zero points on that problem.
- Grading is done as follows.
  Grade 3 = 20–29 points,
  Grade 4 = 30–39 points,
  Grade 5 = 40–50 points.

Good luck!

Paul & Martin

Problem 1. Compare the contents and usage of the MX and SPF records of a DNS server. (3 points)

Answer
The MX-record is a DNS record that specifies to which computer an email to a certain domain should be sent. This computer does not necessarily have to be the final delivery server that delivers the email to the MUA. It can be a firewall, a server shared by several domains used to filter e.g., spam, or it could be a gateway that forwards the email on to another protocol.

Policy Framework (SPF) is a way of determining who is allowed to send emails from a certain domain.

Problem 2. Briefly explain how JavaScript comes into play in

a) AJAX,
b) DNS rebinding attacks,
c) HTTP response splitting attacks. (3 points)

Answer

a) JavaScript XHR used to fetch update from server to update parts of a page.
b) Main page contains JavaScript XHR, providing the victims subsequent GET request(s).
c) Not at all.

Problem 3. Why is it typically a good idea to have random session IDs stored in e.g., cookies? Describe an attack that could work if the session ID is not random. (3 points)

Answer
Session IDs with a sufficiently high level of entropy (randomness) should be used in order to avoid session fixation attacks. Also, cookie-based sessions are better than URL-based sessions in some respects. For example, with cookie-based sessions, the end-user cannot share his session ID by copying the URL from his browser.

Problem 4. DMARC encapsulates and adds funtionality to both DKIM and SPF.

a) What it the purpose of DKIM?
b) Why is DMARC better than DKIM and SPF put together.
c) How does alignment work with DKIM and SPF? (3 points)

Answer

a) DKIM ties a domain name to an email to prevent spoofing attacks. DKIM also provides integrity protection (but not encryption).
b) Both DKIM and SPF are used to authenticate email messages, but they do it in different ways. A limitation to DKIM and SPF is that a sender having implemented DKIM and/or SPF does not know what effect or consequences it has. In the case of DKIM, the sender does not know if there are many emails from the domain that have a bad signature. In the case of SPF, the sender does not know if there is a mistake in the list of allowed IP addresses so that one computer will always fail the SPF check at the receiver side. Another limitation is that the sender does not have any control over what should happen if there is a problem with the DKIM signature and the SPF check fails. One receiver might automatically treat the message as spam while another might let it through without any action. Domain-based Message Authentication, Reporting and Conformance (DMARC) is an effort to standardize a way for senders to announce that they are using DKIM and SPF, to let the sender recommend an action to take if the checks fail, and to let receivers give feedback to the sender.
c) DKIM and SPF uses different identifiers. While DKIM uses the "d=" tag in the signature, SPF uses the domain given in the MAIL FROM command in SMTP.

DMARC has chosen to tie together the identifiers by using the domain given in the "From" header of the message. This header is most often the one used by MUAs and shown to the users. Messages are said to be in alignment if the different identifiers have the same domain. In strict mode the domains have to be identical, while in relaxed mode it is enough that the organizational domain is the same.

Problem 5. Digest authentication (RFC2617) calculates the digest according to

$$\text{MD5}(\ \text{MD5}(A1) : \mathit{nonce} : \mathit{nc} : \mathit{cnonce} : \mathit{qop} : \text{MD5}(A2)\ ),$$

with

$$\begin{aligned} A1 &= \mathit{username} : \mathit{realm} : \mathit{password}, \\ A2 &= \begin{cases} \mathit{method} : \mathit{URI} & \text{if } \mathit{qop} = \mathit{auth}, \\ \mathit{method} : \mathit{URI} : \text{MD5}(\mathit{entity\text{-}body}) & \text{if } \mathit{qop} = \mathit{auth\text{-}int}. \end{cases} \end{aligned}$$

A client request may resemble

```
GET /dir/index.html HTTP/1.0
Host: localhost
Authorization: Digest username="Mufasa",
                      realm="testrealm@host.com",
                      nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
                      uri="/dir/index.html",
                      qop=auth,
                      nc=00000001,
                      cnonce="0a4f113b",
                      response="6629fae49393a05397450978507c4ef1",
                      opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

a) Explain the usage and purpose of the MD5(*entity-body*) part?

b) Explain the usage and purpose of the *cnonce* parameter?

c) Why is Basic Digest Authentication insecure? (3 points)

Answer

a) Provides message integrity with a HMAC-like functionality.

b) A nonce that the client chooses. Prevents time-memory tradeoff attacks.

c) With basic authentication, the user will send username and password in cleartext (Base64-encoded).

Problem 6. Give a regular expression that checks if an input is an email address of a subdomain to either one of the three top-level domains `com`, `se` or `nu`. Assume that the email address only contains alphanumerical characters, where applicable. (3 points)

Answer
One possibility is `^[A-Za-z0-9]+@[A-Za-z0-9]+\.(com|se|nu)$`

Problem 7. Two engineers walk into the Base64 Bar. Engineer A orders "QkVFUg==" and engineer B orders "Sk9MVA==". Who gets to drive the car on the way home?

Hint 1: Only one of the engineers drinks alcoholic beverages.
Hint 2: Decimal representation of ASCII characters is given by:

$$\text{A} = 65, \text{B} = 66, \ldots, \text{Z} = 90, \text{a} = 97, \text{b} = 98, \ldots, \text{z} = 122$$

The Base64 alphabet is:

$$0 = \text{A}, \ldots, 25 = \text{Z}, 26 = \text{a}, \ldots, 51 = \text{z}, 52 = 0, 53 = 1, \ldots, 61 = 9, 62 = +, 63 = /$$

(3 points)

Answer
B does.
Engineer A orders BEER and engineer B orders JOLT.

Problem 8. Explain how Content Security Policy (CSP) is used to prevent XSS attacks. Where is the policy enforced? (2 + 1 points)

Answer
CSP adds HTTP headers to a page, and these headers specify from where resources may be retrieved. The browser must enforce the restrictions specified in the CSP headers provided by the server.

Problem 9. One possible php.ini setting is:

```
allow_url_include = 1
```

Describe the attack that requires this setting. What other conditions must be met for the attack to be possible? (3 points)

Answer
This is a setting that is typical for a remote file inclusion attack. A necessary prerequisite for this attack to be possible is that user input is not filtered correctly, so that an URL pointing to the attacker's external file (on his own server) can successfully be passed as a parameter.

The php.ini setting `allow_url_fopen = 1` is also necessary.

Problem 10. Consider the Domain Name System Security Extensions (DNSSEC).

a) What is the purpose of the NSEC record?

b) Do DNSSEC signatures need to be recalculated between requests from different users? Motivate your answer.

c) Explain one serious negative effect of employing DNSSEC. (3 points)

Answer

a) The NSEC record provides authenticated denial of existence.

b) No, DNSSEC answers are precalculated so that their content does not directly depend on the input.

c) DNS amplification becomes more efficient as the size of the answers increases due to the use of signatures.

Problem 11. Password cracking using TMTO/Rainbow tables.

a) Explain how the chains of a TMTO/Rainbow table (choose one) are traversed when inverting a hashed password. Make sure that you mention the terms *start point, end point* and *reduction function*.

b) Compare the password cracking efficiency in the three cases of using

i) no salt,

ii) a unique salt for the entire site,

iii) a unique salt per user. (2 + 3 points)

Answer

a) If the hashed password is in the table, we essentially reconstruct the corresponding chain. Apply the reduction function to obtain the next password in the chain. If an endpoint has been reached (lookup into end point hash table is $O(1)$), go to corresponding start point. If not, apply hash and reduction function again until an end point is reached. Go to corresponding start point and keep iterating hash and reduction function until the entire chain has been reconstructed. The last password in this process inverts the given hash.

b) Without salt, we can most likely download pre-built TMTO tables for efficient cracking without having to construct the tables ourselves. We can target all user passwords with the same table in this case. With a site-wide salt, we need to construct the TMTO tables. This will cost us a pre-processing time that is equal to the entire search space. However, once this/these tables are constructed, we can use it/them to recover the passwords for all users. Using a unique salt for every user renders TMTO attacks obsolete. A new table would need to be constructed for every salt (and password), so a brute-force approach is a better option.

Problem 12. Consider a Hashcash solution in which a string

$$ver : bits : date : resource : rand : counter$$

is hashed using SHA-1, where

ver is version number (currently 1),
bits indicates how costly the function is for sender,
date gives current date,
resource is recipients email address,
rand is a random number.

A spammer plans on including a Hashcash header with each mail she sends.

a) How is a *valid* Hashcash header with $bits = 30$ constructed?
b) How many calls to SHA-1 does it take to *generate* a Hashcash header with $bits = 30$? Exactly or on average?
c) How many calls to SHA-1 does it take to *verify* a Hashcash header with $bits = 30$? Exactly or on average?
d) Why are Hashcash headers with $bits = 80$ and $bits = 1$ impractical?
e) What is the purpose of the *rand* parameter? (5 points)

Hint: If x is a randomly chosen input and $h = \text{SHA-1}(x)$ is the corresponding 160-bit hash, then every bit position in h has value 0 or 1 with probability $\frac{1}{2}$.

Answer

a) Choose *rand* randomly and set $counter = 0$. Increment *counter* until the SHA-1 hash of the string has 30 initial zero bits.
b) 2^{30} times on average.
c) Exactly once.
d) For $bits = 80$, it is impossible to generate with reasonable resources, hashing about 2^{80} times. For $bits = 1$, it is not very costly to generate, so there is virtually no penalty for the sender, defying the purpose of it.
e) Separates different senders. The client must store previously used headers so that they are not reused.

Problem 13. A DNS cache poisoning attack can be very valuable if it is successful.

a) Explain how a DNS cache poisoning attack works.
b) How should DNS queries be constructed in order to minimize the success probability of the attack?
c) How would the attack be affected if queries were sent using TCP instead of UDP?

(5 points)

Answer

a) A DNS cache poisoning attack aims to add or change DNS records on servers so that the wrong answer will be sent to clients. If the attacker can control the IP addresses sent as response to queries, then traffic can be directed to an attacker's computer. The lecture notes describe two variants. One simple variant is to send false responses to queries, sending additional DNS results for `www.bank.com` when only `www.attacker.com` was asked for. A more interesting and advanced variant is the attack illustrated in Figure 6 in the lecture notes (preferred answer).

b) The transaction ID and port number should be chosen uniformly at random for maximum entropy.

c) The attack would not work at all. A successful attack would require IP spoofing, which is practically impossible due to the TCP handshake.

Problem 14. Briefly explain the following terms.

a) Greylisting

b) DS record

c) Same-origin policy

d) SMTP

e) DNS rebinding (5 points)

Answer

a) A spam prevention technique that temporarily rejects a mail if the sender is previously unknown. Circumvented by following the protocol and resending, after which the mail is immediately accepted.

b) Delegation signer, DNSSEC record, stores hash of DNSKEY for child domain.

c) Browser-enforced policy that restricts information sharing between web pages.

d) Simple Mail Transfer Protocol.

e) An attack in which the attacker's DNS server changes its responses for subsequent calls in order to trick a client's browser to violate the same origin policy.
