<u>Quiz 1:</u> Next Wednesday, Walker, during regular class time

- L1: Security Basics
- L2: Baggy Bounds
- L3: Hacking Blind
- L4: OKWS
- L5: Guest Lec.
- L6: Capsicum & Conf. Deputy
- L7: NaCl
- L8: OWASP & Tangled Web
- L9: Django & CSRF
- L10: Klee & Symbolic Exec.
- L11: Ur/Web
- L12: TCP/IP
- L13: Kerberos
- L14: ForceHTTPs
- --
- Lab1: Buffer Overf.
- Lab2: Priv. Sep.
- Lab3: Concolic Exec.

How to study for Quiz 1

- Do old exams.
- Review papers and lecture notes together.
- Youtube videos available for each lecture.
- Student questions and lecture questions for each lecture are available on the submission page.

L1: Intro

Policy

- the goal you want to achieve
- negative goals (e.g. “system should not crash”)
- CIA (confidentiality, integrity, availability)

Threat model

- attacker’s capabilities (access to source? physical access? etc.)

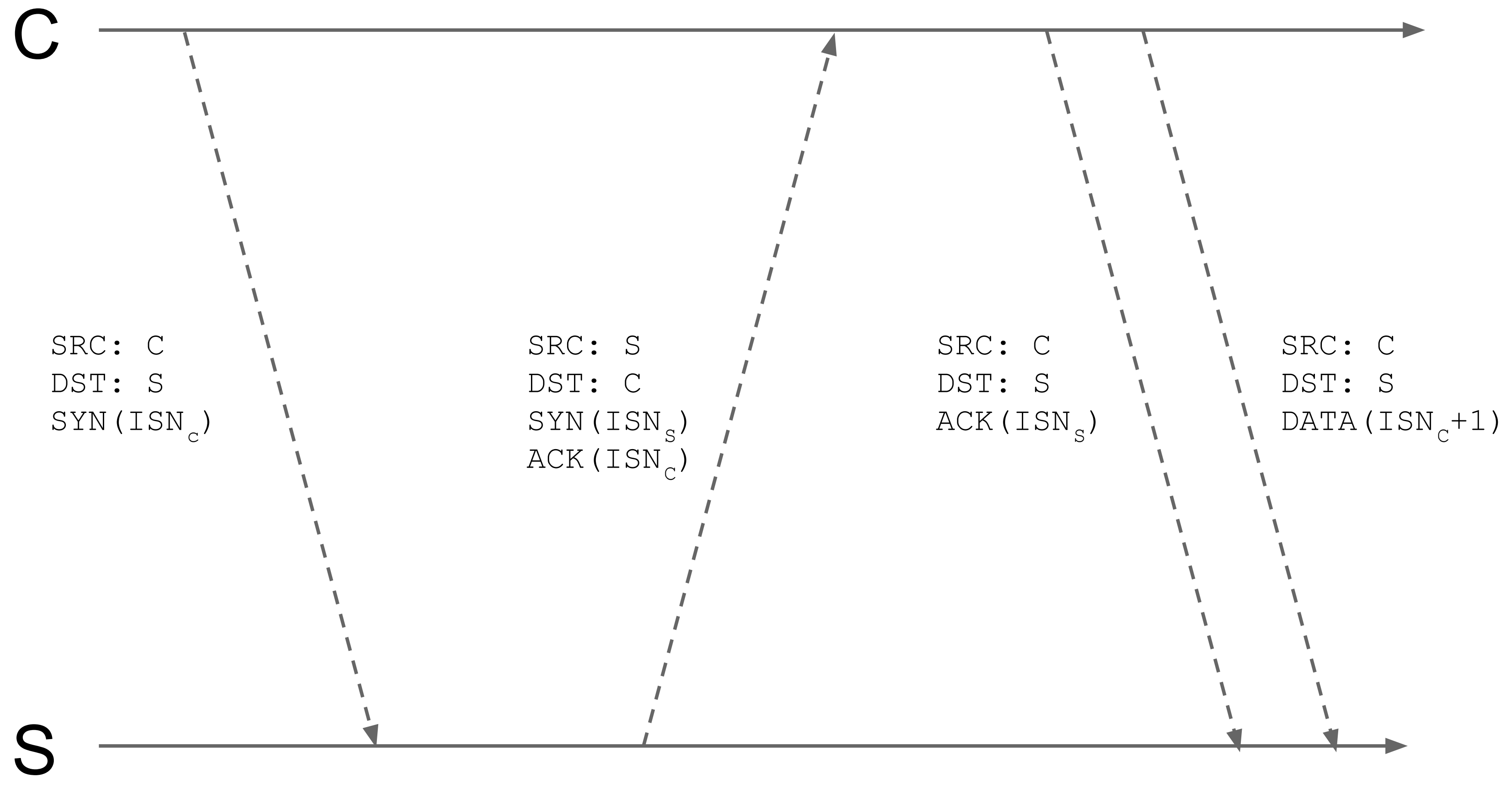
Mechanism

- technology to enforce policy (unix permissions, capabilities, etc.)

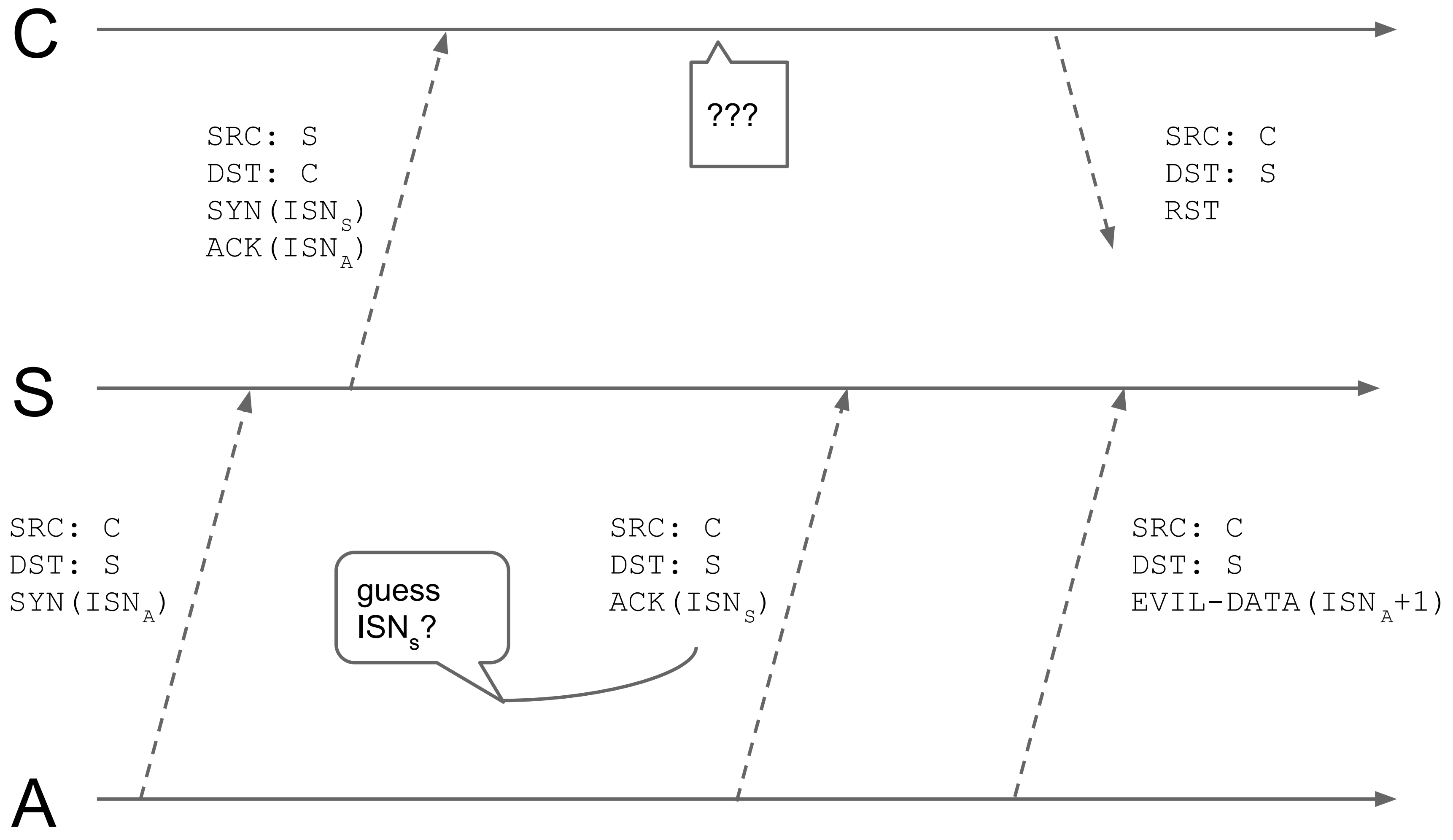
L12: TCP/IP and Network Security

1. TCP Handshake
2. Spoofing connections
3. DOS attacks (RST, SYN Flooding)
4. DNS, ARP, DHCP, BGP, ...
5. Old Quiz Problems

[L12, 1/5] TCP Handshake



[L12: 2/5] Connection Spoofing



[L12: 2/5] Connection Spoofing

Problem

- Easy for A to guess ISN_s
    - $ISN_s \leftarrow (ISN_s)_{old} + 250000 * \Delta t_{sec}$
    - A can get $(ISN_s)_{old}$ by sending a legal $SYN(ISN_A)$ packet to S.

Solution

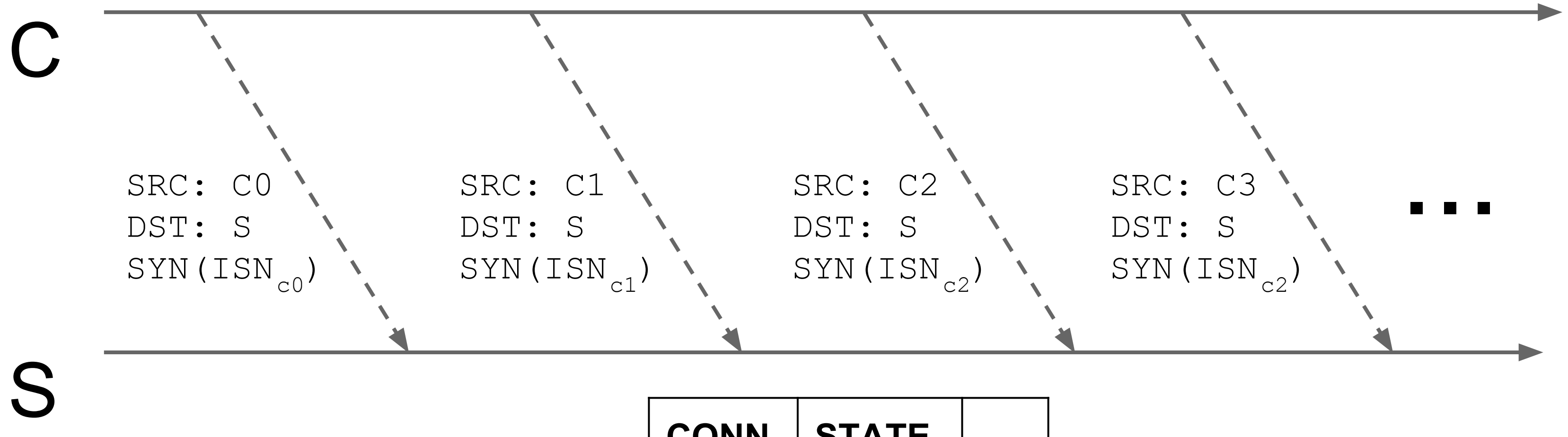
- Make it hard for A to guess ISN_S
    - $ISN_s \leftarrow ISN_{oldstyle} + F(ipaddr_{src}, portn_{src}, ipaddr_{dst}, portn_{dst}, key)$
    - F a cryptographic hash function (e.g. SHA1)

[L12: 2/5] Connection Spoofing

- If A knows SN_C:
  - Reset connection: RST($SN_{C\mp\square}$)
  - Inject data into existing stream: DATA(SN_C)

[L12: 3/5] DOS Attacks

SYN Floods



CONN	STATE	...
C0	waiting	
C1	waiting	
C2	waiting	
...	...	

[L12: 3/5] DOS Attacks

Problem with SYN Floods

- Server must keep state for every incoming SYN packet

Solution: SYN Cookies

- Don’t keep state for new incoming SYN packets!
    - Reply w/ SYN-ACK as normal.
- $\boxed{ISN_s \leftarrow F_{key}(ipaddr_{src}, portn_{src}, ipaddr_{dst}, portn_{dst}, timestamp) \mathbin{||} timestamp}$
    - timestamp on scale of minutes

[L12: 4/5] DNS, ARP, DHCP, BGP

See lecture notes and paper for:

- DNS response spoofing
- DNS amplification
- ARP spoofing
- DHCP spoofing
- others

[L12 5/5] Quiz 1, 2013, Problem 6

Ben Bitdiddle tries to fix the Berkeley TCP/IP implementation, described in Steve Bellovin's paper, by generating initial sequence numbers using this random number generator:

```
class RandomGenerator(object):
    def __init__(self, seed):
        self.rnd = seed
    def choose_ISN_s(self):
        isn = self.rnd
        self.rnd = hash(self.rnd)
        return isn
```

Assume that Ben's server creates a RandomGenerator by passing it a random seed value not known to the adversary, that hash() is a well-known hash function that is difficult to invert, and that the server calls choose_ISN_s to determine the ISNs value for a newly established connection.

How can an adversary establish a connection to Ben's server from an arbitrary source IP address, without being able to snoop on all packets being sent to/from the server?

L13: Kerberos

1. Overview
2. Practice problems

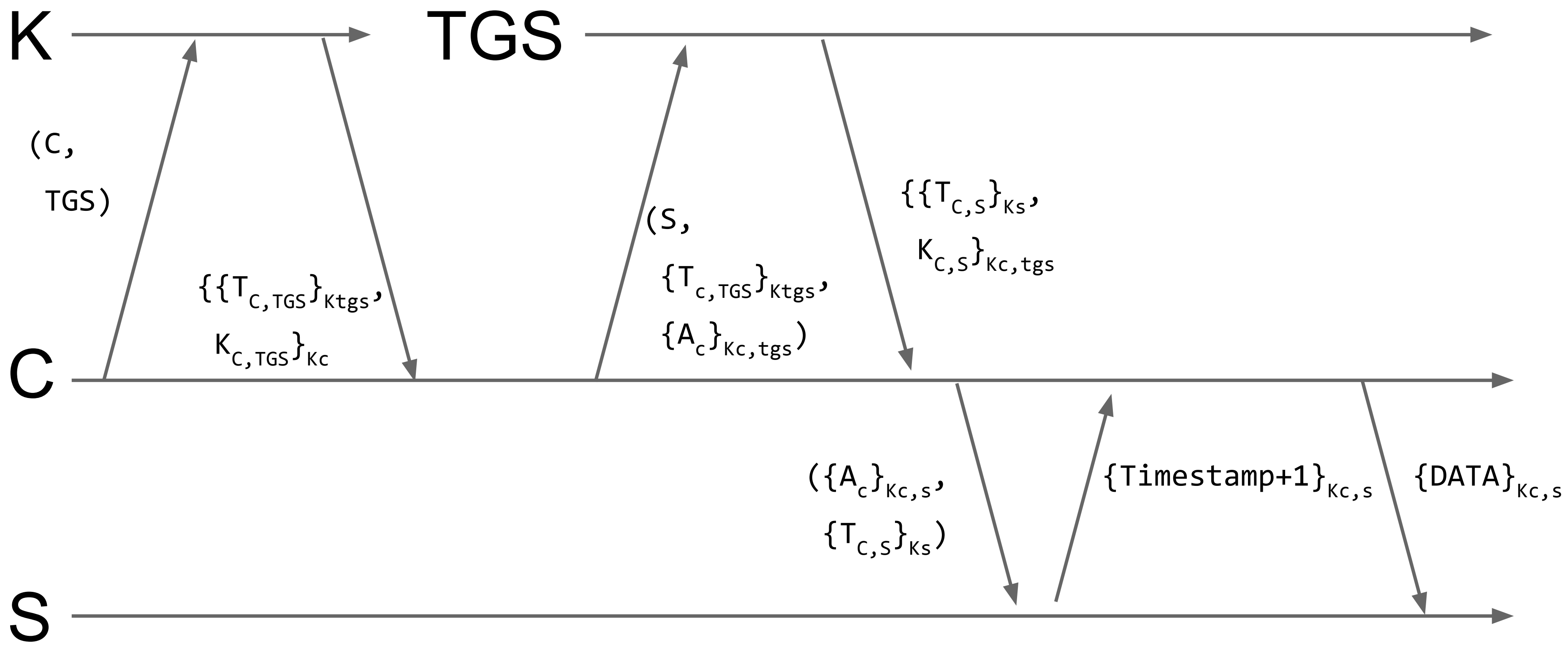
For fun, see:

http://web.mit.edu/kerberos/dialogue.html

L13: Kerberos Overview

$T_{c,s}$ = {s, c, addr, timestamp, life, $K_{c,s}$ }

A_c = {c, addr, timestamp}$_{Kc,s}$


K
TGS
C
S
(C, TGS)
{{T$_{C,TGS}$}$_{Ktgs}$, K$_{C,TGS}$}$_{Kc}$
(S, {T$_{c,TGS}$}$_{Ktgs}$, {A$_c$}$_{Kc,tgs}$)
{{T$_{C,S}$}$_{Ks}$, K$_{C,S}$}$_{Kc,tgs}$
({A$_c$}$_{Kc,s}$, {T$_{C,S}$}$_{Ks}$)
{Timestamp+1}$_{Kc,s}$
{DATA}$_{Kc,s}$

L13: 2009 Quiz 2 Problem 2

Bob logs into an Athena workstation, which uses Kerberos to obtain a ticket for bob@ATHENA.MIT.EDU, and then runs Bob’s mail client, which contacts Bob’s post office server to fetch new messages.

2. Alice doesn’t want Bob to know about an upcoming event, which was announced to Bob via email. To this end, Alice plans to intercept Bob’s communication with his post office server, and to pretend that Bob has no new mail. Alice can observe and modify all network packets sent by Bob. How does Kerberos prevent Alice from impersonating Bob’s mail server? Be as specific as possible; explain how Bob can tell between Alice and the real mail server in terms of network packets.

L13: 2009 Quiz 2 Problem 3

Now, Alice wants to read Bob’s email, and intercepts all network packets ever sent and received by Bob’s workstation (which is the only computer that Bob uses). However, Alice does not know Bob’s password to access Bob’s post office server, and Bob’s packets to and from the post office server are protected by Kerberos.

3. Suppose that after Bob reads and deletes all of his mail, Alice learns what Bob’s password was. Describe how Alice can obtain Bob’s past messages.

L13: 2012 Quiz 1 Problem 6

Ben Bitdiddle is designing a file server that clients connect to over the network, and is considering using either Kerberos (as described in the paper) or SSL/TLS (without client certificates, where users authenticate using passwords) for protecting a client's network connection to the file server. For this question, assume users choose hard-to-guess passwords.

6. Would Ben's system remain secure if an adversary learns the server's private key, but that adversary controls only a single machine (on the adversary's own home network), and does not collude with anyone else? Discuss both for Kerberos ~~and for SSL/TLS~~.

L13: 2013 Quiz 1 Problem 7

In a Unix Kerberos implementation, each user’s tickets (including the TGT ticket for the TGS service) are stored in a per-user file in /tmp. The Unix permissions on this file are such that the user’s UID has access to that file, but the group and others do not.

7. Ben Bitdiddle wants to send an email to Alyssa, and to include a copy of the Kerberos paper as an attachment, but because he stayed up late studying for this quiz, he accidentally sends his Kerberos ticket file as an attachment instead. What can Alyssa do given Ben’s ticket file? Be precise.

L13: 2013 Quiz 1 Problem 8

8. Ben Bitdiddle stores his secret files in his Athena AFS home directory. Someone hands Alyssa P. Hacker a piece of paper with the key of the Kerberos principal of all-night-tool.mit.edu, which is one of the athena.dialup.mit.edu machines.

   Could Alyssa leverage her knowledge of this key to get access to Ben's secret files? Assume Alyssa cannot intercept network traffic. Explain either how she could do so (and in what situations this might be possible), or why it is not possible.

[L13] SSL/HTTPs/ForceHTTPs

- Covered in lecture next Monday

- HTTPs handles authentication and encryption for web

- Still problematic in some cases
  - ForceHTTPs fixes some browser quirks

6.858 Quiz Review

Blind ROP

6.858 Quiz Review

OWASP/Tangled Web

Security Risks

- Open Redirectors
- Vulnerable Software
- CSRF/XSRF
- Function Level Access Control
- Data Exposure

Security Risks

- Misconfigurations
- Insecure Direct Object Reference
- XSS
- Broken Session Management
- Injections

Same Origin Policy

- Dependent on Protocol and Host
- Blocks Execution and Access
- Not foolproof
  - Java
  - Flash

Same Origin Policy

- Dependent on Protocol and Host
- Blocks Execution and Access
- Not foolproof
  - Java
  - Flash

Content Security Policy

- Specify Constraints on Loading
- Prevent “malicious” scripts
- Still Vulnerable

OKWS

- Separate each service into a distinct process
- Each service runs as a sepreate UID and GID
- Each service is chrooted
- Database proxy is used to enforce query structure

OKWS

- Okld launches all services, creates sockets, run as root
- No protection against XXS
- Pubd stores all templates
- Oklogd keeps log of all activities, chroot to its own sandbox

DB Proxy

- Each service is given a subset of possible queries it is allowed to make
- Each service passes a token to the db proxy with it’s query that proves its identity

DAC

- Each object has a set of permissions
- Applications set permissions on objects
- Privileges are checked when a program access an object

DAC Problems

- Only Root can create new users
- Sometimes hard to customize permissions
- Some objects don’t have clear configurable access control list

MAC

- Users can’t change the policy
- Tries to enforce military classified levels
- Policy is separated from application code

MAC Problems

- The system controls who can access a newly created object
- Hard to customize permissions

Capabilities

- Token based access
- Can have different tokens for read vs write
- Each object accessed by the file handler
- Tokens are passed down to forked processes

Caps Mode

- Once a process enters it can’t leave caps mode
- Name space is restricted, can’t use “..”
- Unix permissions still apply
- Allowed operations stored in file descriptor

Capabilities

Advantages:

- Any process can create a new sandbox (even a sandbox)
- Very easy to customize access

Disadvantages:

- No global namespace
- Hard to keep track of who has access to persistent files

UrWeb

- Only one programming language is used for the entire application
- All objects passed between different parts of the application are strongly typed
- Instead of Http requests, clients call typed functions that are run on the server atomically

Strongly Typed

Strongly typed objects are objects that can only be used in specific functions of the application. This prevents XXS because string input from the user cannot be transformed into HTML or javascript unless the author of the application explicitly allows it.