Engineering Robust Server Software

Defense In Depth
You Are Building YourAwesomeSite.com

- Use all the best practices you know
You Are Building YourAwesomeSite.com

...But also lots of things you didn't write
  • Adds a lot of complexity…
You Are Building **YourAwesomeSite.com**

- **Hey Beavis,**
  I found this code on StackOverflow

- I don't know but they said it's really awesome.

- **Heheh What's it do Butthead? Heheh**

- **Oh plus what about the other developers on your team?**
What Happens If Something Goes Wrong?

- Suppose a vulnerability exists: what is the damage?
Defense In Depth

- Idea: Assume one layer of security might fail
  - Multiple layers of security
  - Minimize damage if one layer is compromised
Example of This That We Have Seen?

- What have we already seen that is an example of mitigating damage if compromised?
  - A: nop slide
  - B: CSRF token
  - C: Diffie-Hellman Key Exchange
  - D: Salt and hash passwords
Famous Example of NOT Defense In Depth

- Equifax got hacked
  - Bug in web library they were using
  - Many users' personal data (SSNs, etc) stolen
- Why/how?
- What should they have done?
Vulnerability -> Access To All Things

SELECT * from userssns;

Django

Built In Authen
Sanitization
Distrust clients

AwesomeThingyLib
ReallyNeatWhatzit
BlahBlahToolkit
What Could We Do Instead?

Users

Web Code

Internal APIs

CC Server

Auth Server

SSN Server
What Could We Do Instead?

Users

Web Code

Internal APIs

CC Server

Auth Server

SSN Server

+Firewalls
Contrast: Make a Purchase

Users

Web Code

Auth Info

CC Info

....
Contrast: Make a Purchase

GET /selectPayment

(Payment page)

Auth Info
CC Info
....

Users

Web Code

Authen info

Payment Info
Contrast: Make a Purchase

Users → Web Code

GET /selectPayment

(Payment page)

POST /makePayment

To CC Processor

Auth Info
CC Info

....

Authen info
Payment Info
Contrast: Make a Purchase

GET /selectPayment

Auth Info
CC Info
....

Users

Web Code

Authen info

Get all user's info
(all the info)

(everyone's credit card #s
hashed pwds
SSNs...)

GET /selectPayment

Authen info

Get all user's info
(all the info)

(everyone's credit card #s
hashed pwds
SSNs...)

15
Contrast Make A Purchase

GET /selectPayment

Web Code

CC Server

Auth Server

SSN Server
Contrast Make A Purchase

GET /selectPayment

Web Code

isSessionValid?

CC Server

Auth Server

SSN Server
Contrast Make A Purchase

GET /selectPayment

Web Code

Auth Server

CC Server

SSN Server

Yes
 Contrast Make A Purchase

GET /selectPayment

Get Cards for user=brian sid=123456789
Contrast Make A Purchase

GET /selectPayment

Get Cards for user=brian
   sid=123456789

IsValid?
   Yes

Web Code

CC Server

Auth Server

SSN Server
Contrast Make A Purchase

GET /selectPayment

Web Code

Get Cards for user=brian
sid=123456789

ending in 9876
ending in 0000

CC Server

Auth Server

SSN Server

Only give back what is needed
to web server!
(e.g., not full credit card #s)
Contrast Make A Purchase

Purchase $579, cardid=1, user=brian...

POST /makePayment

IsValid?
Yes

Web Code
CC Server
Auth Server
SSN Server
Contrast Make A Purchase

POST /makePayment

Web Code

CC Server

Auth Server

Bank's Server

SSN Server
POST /makePayment

No need to ever report full credit card numbers back to web code (only time it sees is when user adds them)
Contrast Make A Purchase

Attacker can only ask other services to do stuff based on well-defined APIs.
No direct access from outside world: attacker can’t hit this bug directly, probably can’t hit it indirectly.
Contrast Make A Purchase

Attacker probably needs these TWO bugs! Less likely.
Let Us Revisit This

Web Code

CC Server

Auth Server

SSN Server

I'm going to play a longer game...
Let Us Revisit This

Every time someone logs in
- Get their auth info
- Send request to CC server
  Purchase something with their card

Web Code

CC Server

Auth Server

SSN Server

Less bad, but still bad... (why?)
Let Us Revisit This

Every time someone logs in:
- Get their auth info
- Send request to CC server
Purchase something with their card

How to defend against this?
Remember this "plan"?

- **Most secure:**
  - Run program Handle web request on a computer
  - Throw away computer
  - Buy new computer
  - Run next program Handle next web request on it

$$$$$
Ok that plan was bad... but

• That plan was bad, but what did we decide we could do instead?

Containers.
Prevention + Detection + Response

• So far have talked about **prevention**
  • Keep bad things from happening
  • Reduce badness if they do happen

• Also want **detection**
  • Know when a bad thing has happened / is happening

• …and to be able to **respond** to the attack
  • Nice if we can do something about it…
Intrusion Detection

- Monitor system for suspicious activity

SYN, port 22
(encrypted traffic)
FIN

Seems normal.
Some short ssh activity
Intrusion Detection

- Monitor system for suspicious activity

SYN, port 22
(encrypted traffic)
FIN

Again... Odd, that's two in quick succession...
Intrusion Detection

- Monitor system for suspicious activity

SYN, port 22
(encrypted traffic)
FIN

Hmm maybe they are trying passwords?
Seems bad
Intrusion Detection

- Monitor system for suspicious activity

SYN, port 22 (encrypted traffic)
FIN

Ban this IP in the firewall

Response
Was this response good?

- Detected something suspicious
- Responded strongly:
  - Blocked traffic from originating site
- Good or bad?
Was this response good?

• Detected something suspicious
• Responded strongly:
  • Blocked traffic from originating site
• Good or bad?
  • It depends!
Intrusion Detection

- If true positive, outcome was good
Intrusion Detection

- If false positive, then it was bad
  - Abnormal does not always mean evil

I was just trying to scp several small files one at a time...

Ban this IP in the firewall

FIN
Detection

- **Automated**: Algorithmic analysis + detection
  - Signature based: look for patterns
    - This seems to be trying many passwords
    - This seems to be port scanning
  - Anomaly detection:
    - Develop ML model of normal behavior
    - Find things that deviate

- **Human**:
  - Look at logs, system behavior etc
Detection

• Not limited to network activity
  • These aren't queries that we ever run…
  • This return address has been overwritten
  • This pattern of system calls is unusual
  • There have been 4 failed login attempts for user “brian"
  • …

• Similar ideas in non-computer security
  • Bank watches credit card purchases for suspicious activity
  • Unattended bags at airport
  • …
Responses

- Notify administrators
  - **Send email:** Hey something is strange… Here is what is up!
  - Pros and cons?
Responses

• Notify administrators
  • Send email, text, etc: Hey something is strange… Here is what is up!
  • Pros and cons?

• Block suspicious behavior
  • Lock account, firewall traffic, …. 
  • Pros and cons?
Responses

• Notify administrators
  • Send email, text, etc: Hey something is strange… Here is what is up!
  • Pros and cons?

• Block suspicious behavior
  • Lock account, firewall traffic, ....
  • Pros and cons?

• Shutdown affected system
  • Power that machine off
  • Pros and cons?
Responses

• Notify administrators
  • Send email, text, etc: Hey something is strange… Here is what is up!
  • Pros and cons?

• Block suspicious behavior
  • Lock account, firewall traffic, …. 
  • Pros and cons?

• Shutdown affected system
  • Power that machine off
  • Pros and cons?

• Nuke and restore from backup? (or even throw away hardware?)
Factors in Choosing Response

- False positive rate
  - How certain are we that suspicious = bad?
- Severity of suspected attack
  - How bad is it?
    - Someone trying to find a vulnerability vs
    - Server was rooted
- Impacts of response on "good" users/ how many affected
  - Bad impacts: services temporarily unavailable, …
  - Good impacts: prevent leakage of sensitive info,…
Wrap Up

• Assume security measures will fail!
  • Multiple levels: mitigate damage if one fails

• Detect suspicious activity
  • Don't just assume everything is good, look for bad stuff

• Respond to threats
  • What to do: it depends…