# Lecture 15 - Web Security

CSE497b - Spring 2007

Introduction Computer and Network Security

Professor Jaeger

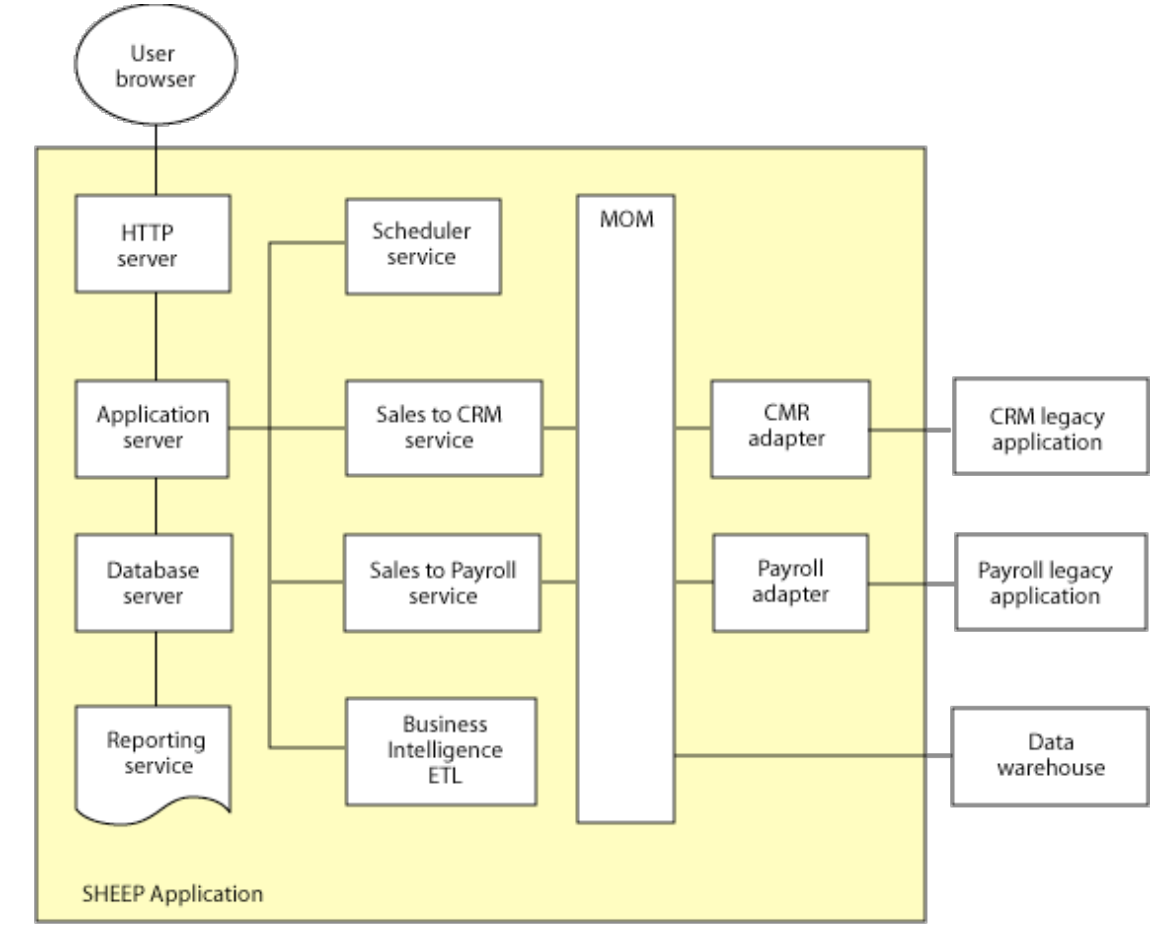
www.cse.psu.edu/~tjaeger/cse497b-s07/

# Web Server

- Entry point for clients
  - To a variety of *services*
  - *Customized* for clients (e.g., via cookies)
  - Supported by *complex backend applications* (e.g., databases)
- Target of attackers
  - *Common protocol*
  - Supports a *wide range of inputs*
  - *Complex* software *interactions*
  - Running with *high privilege*
- Q: How does this impact?
  - Vulnerabilities, Threats, Risks

# Web Server Deployments

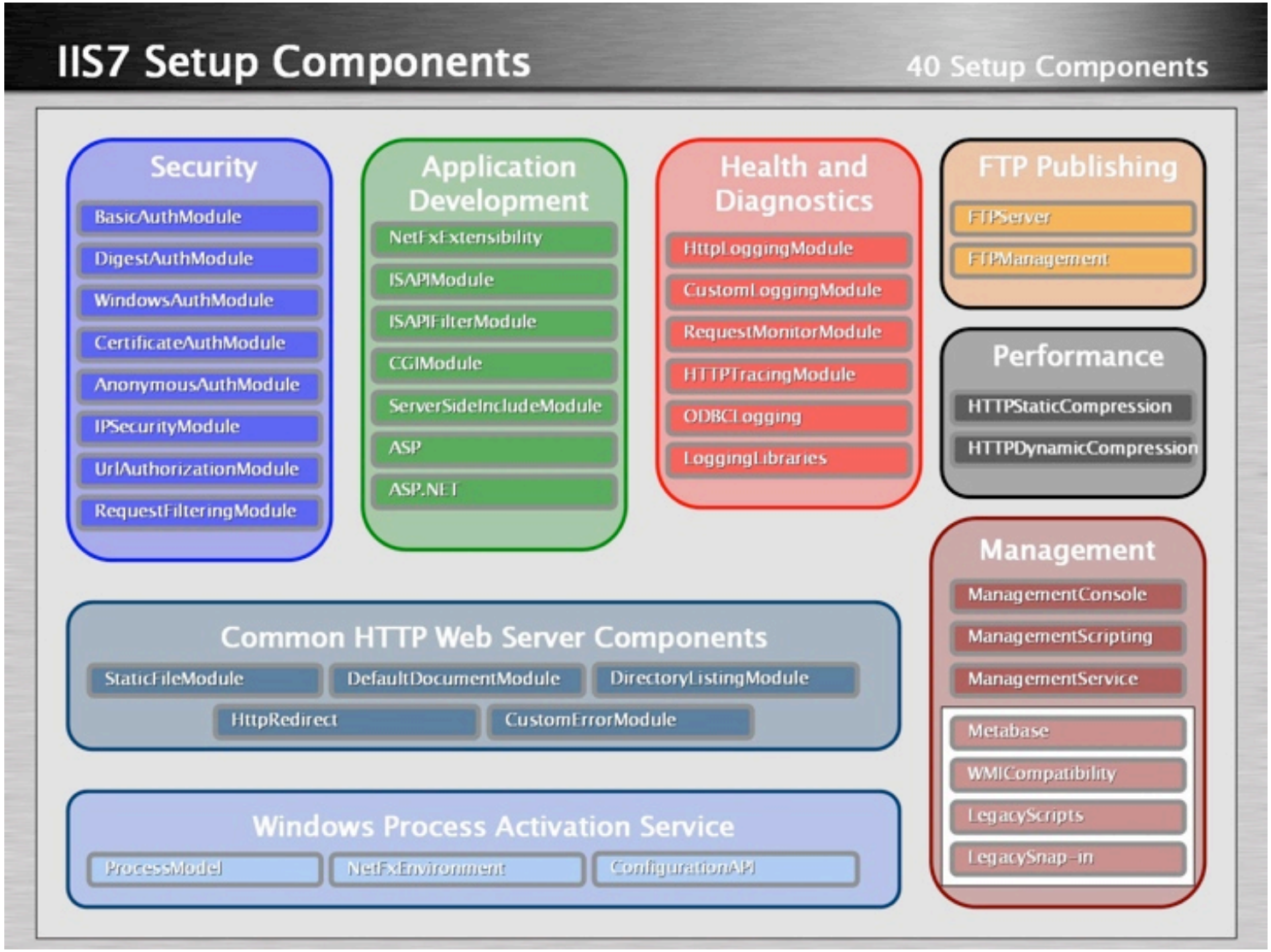
- Note the multiple application layers and connection to legacy code




MOM = Message-oriented middleware
CRM = Customer relationship management
ETL= Extract Translate Load

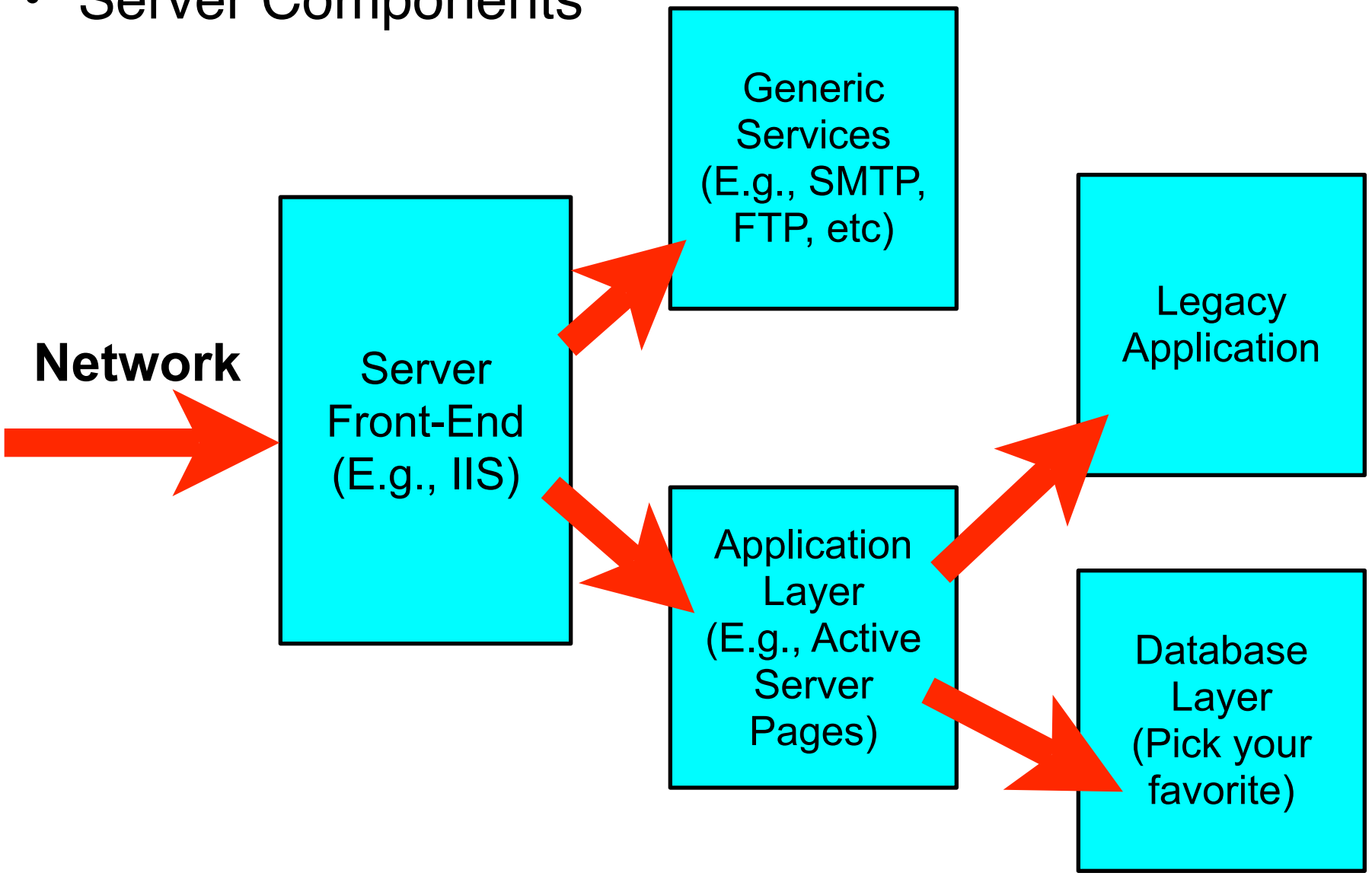
# Web Server Software

- E.g., IIS 7



# Web Server Architecture

- Server Components



# Server-side Scripting

- Program placed directly in content, run at during request time and output returned in content
  - MS active server pages (ASP)
  - PHP
  - mod_perl
  - server-side JavaScript
  - python, ....
- Nice at generating output
  - Dangerous if tied to user input

# Dynamic Content Security

- Largely just applications
  - Inasmuch as application are secure
  - Command shells, interpreters, are dangerous
- Three things to prevent DC vulnerabilities
  - Validate input
    - Input often received as part of user supplied data
    - E.g., cookie
  - Limit program functionality
    - Don't leave open ended-functionality
  - Execute with limited privileges

# Web Server Vulnerabilities

- Not surprisingly, these are numerous
- For IIS 5, focus was on function
  - All services were ON by default
  - Buffer overflow -- e.g., Code Red
- Interactions between components are complex
  - HTTP input to database queries
  - SQL Injection -- execute user input directly
- Web server permissions
  - Web servers have broad access
  - Deface web server -- modify server files
  - Compromise system -- modify system files

# What can be done?

- Checklist for IIS 5
  - windows.stanford.edu/docs/IISseccheckIist.htm
  - Gives an idea of what must be done for IIS
- Some examples
  - "Disable all unnecessary ISAPI filters [services]"
    - "Delete DLLs [libraries] associated with disabled filters"
  - "Website must never be on the system drive"
  - "Only necessary services" -- only SMTP
  - "Remove NTFS write permissions where possible"
  - Obscurity
    - "Don't use obvious names for script and code directories"
    - "Set default website to extreme security"
- IIS 7 does does many of these -- automate all?

# Web Server as a Host Security Problem

- Adversary's Goal
  - Integrity/Secrecy/Availability
  - Get code running on your system
    - That is under the adversary's control
- Ways to Execute Code
  - Accessible interfaces
    - Defense: minimize attack surface
  - Vulnerable interfaces
    - Defense: prevent various code injections: buffer overflows
- Privilege
  - Attackers want this code to do as much as possible
    - Defense: minimize its privilege

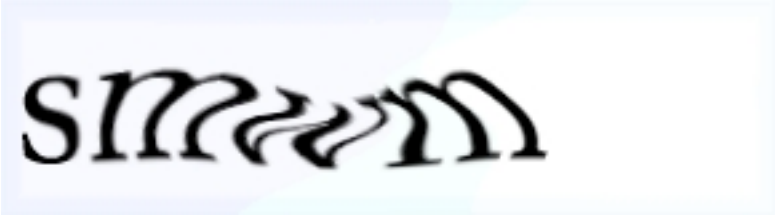
# Canonical (common) DOS - Request Flood

- Attack: request flooding
  - Overwhelm some resource with legitimate requests
  - e.g., web-server, phone system


- Note: unintentional flood is called a *flash crowd*

# DOS Prevention - Reverse-Turing Tests

- *Turing test*: measures whether a human can tell the difference between a human or computer (AI)
- *Reverse Turning tests*: measures whether a user on the internet is a person, a bot, whatever?
- CAPTCHA - **c**ompletely automated public Turing test to tell computers and humans apart
  - contorted image humans can read, computers can't
  - image processing pressing SOA, making these harder



- Note: often used not just  for DOS prevention, but for protecting "free" services (email accounts)

# DOS Prevention - Puzzles

- Make the solver present evidence of "work" done
  - If work is proven, then process request
  - Note: only useful if request processing significantly more work than
- Puzzle design
  - Must be hard to solve
  - Easy to Verify
- Canonical Example
  - Puzzle: given x-bits of input r and h(r), where h is a cryptographic hash function
  - Solution: Invert h(r)
  - Q: Assume you are given 108 bits of input for 128-bit hash input, how hard would it be to solve the puzzle?

# Take Away

- The complexity of web server (and web client) systems makes ensuring their security complex
  - A single interface (HTTP) enhances function
  - Lots of services can be accessed which makes attack surface large
  - The variety of inputs via this interface makes detecting malicious input very difficult
  - Privileges available to injected code can be sufficient to take over system
- Servers are high profile targets
  - Valuable info (credit cards, private user data)
  - Represent an entity (denial of service)