Web Timeouts and Their Implications

Zakaria Al-Qudah* – CWRU

Michael Rabinovich – CWRU

Mark Allman – ICSI

April 9, 2010

*Now with Yarmouk University, Jordan

Timeouts and Claim-and-Hold Attacks

- Timeouts are fundamental to the operation of network protocols
- Timeouts can be utilized to mount claim-and-hold DoS attacks
  - Attacker claims server's resources and maintains minimal level of interactivity with the server to keep them held
- Long timeouts help tolerating legitimate anomalies, but make claim-and-hold attacks easier
- Short timeouts raise the bar for the attacker, but limit the tolerance to legitimate anomalies.

Questions:

- What are the values of various timeouts in today's Web servers?
- How these timeouts relate to the characteristics of actual Web transactions?
- How Web servers can implement timeouts to better cope with possible attacks?

Measurements Description

- Probe operational Web server for various timeouts
  - Top 500 high volume sites (from Alexa.com)
    - 53% reported some version of Apache, 12% Microsoft IIS, 10% GWS (google), the rest others/unknown
  - 15K regular sites collected using a link harvester tool
    - 68% Apache, 10% Microsoft IIS, the rest others/unknown
- Compare results to actual times needed by Web clients
  - A week-long packet trace of Web traffic from ICSI
  - Captured on August 2009
  - Contains nearly 1.6M HTTP connections involving nearly 14K servers and 25K clients

Considered Timeouts

- Application Timeout: Time from completing connection establishment to receiving the first byte of the HTTP request
- Request Timeout: Time from receiving first byte of the HTTP request to receiving last byte of the request
- Response Timeout: Time for the client to consume the entire response
- TCP Timeout: Time for the server to receive acks from the client for its data packets
- HTTP Keep-Alive Timeout: Time for a server to receive a subsequent HTTP request after finished serving the current request on a persistent HTTP connection

Application Timeout

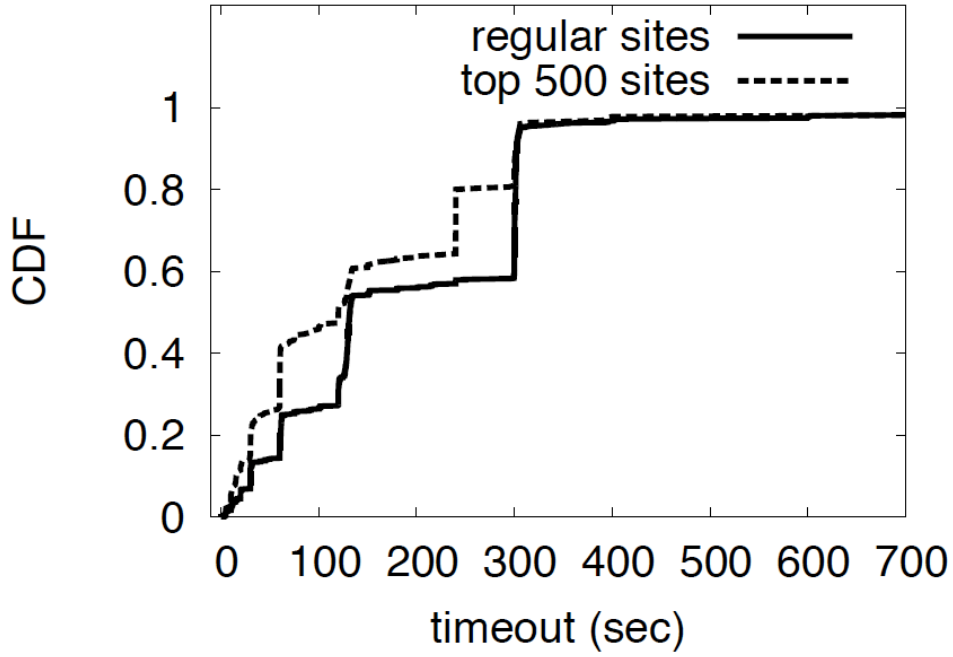
- The time the server allows from completing the connection establishment to receiving the first byte of the HTTP request
- Operational timeout
  - Open connection, observe when the connection is closed
- Actual transactions
  - Measure the time from the last ack of TCP 3WHS to the first packet of HTTP request

Results:

- The transmission of 99 % of the requests has started within 1 second of connection establishment

Results:

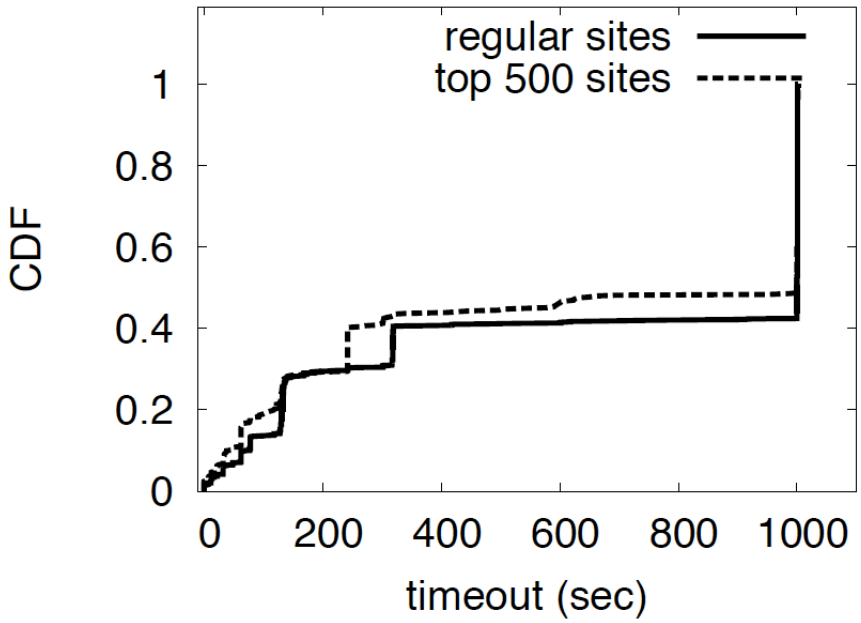
- Nearly 36% of sites do not terminate the connection after 20 min of waiting (TCP_DEFER_ACCEPT)
- Application timeout does not apply to these servers



Request Timeout

- Time from receiving first byte of the HTTP request to receiving last byte of the request
- Focus on HTTP GET requests
- Operational timeout
  - Open connection, send 1000-bytes HTTP GET request, one byte/sec, and observe if (or when) the connection is terminated
- Actual transactions
  - Measure time between the client sending first and last packets of an HTTP connection

Results:



Results:

- 85% of requests fit into 1 packet
- 99.9% of requests take less than 1 second to be transmitted

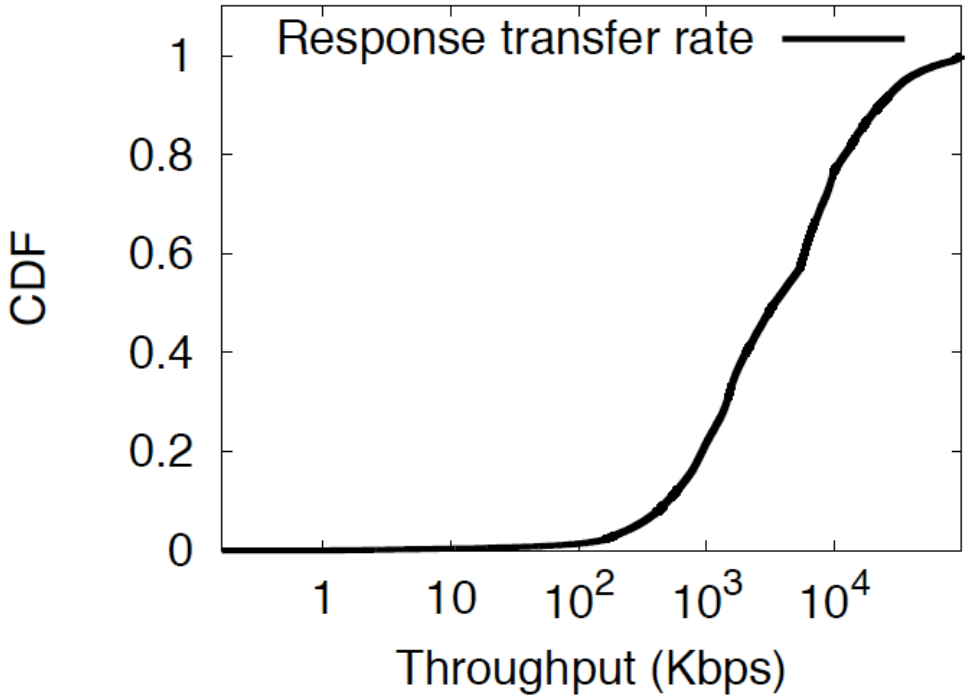
Response Timeout

- Time for the client to consume the entire response
- Operational timeout
  - Open a connection, send a request, consume the response at 100 bytes/sec
  - Observe if the server terminates the connection
  - Reasoning: we are aware of only one major Web server that impose a response timeout (Microsoft IIS, default 240 bytes/sec)
- Actual transactions
  - Measure response transfer rate for large responses

  (> 50KB)

Results:

- Only about 24% of sites impose a limit on response rate

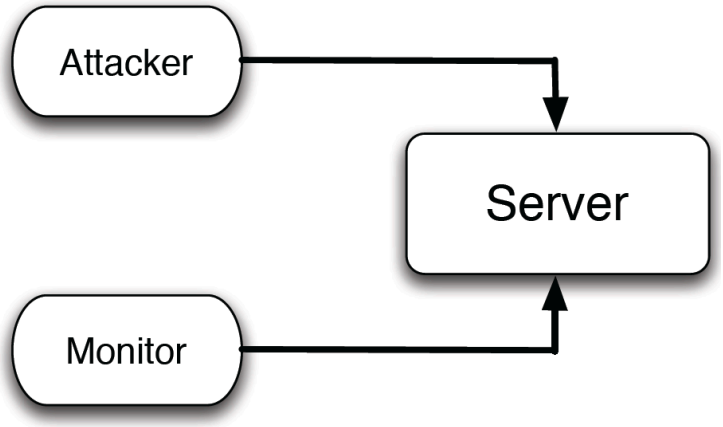


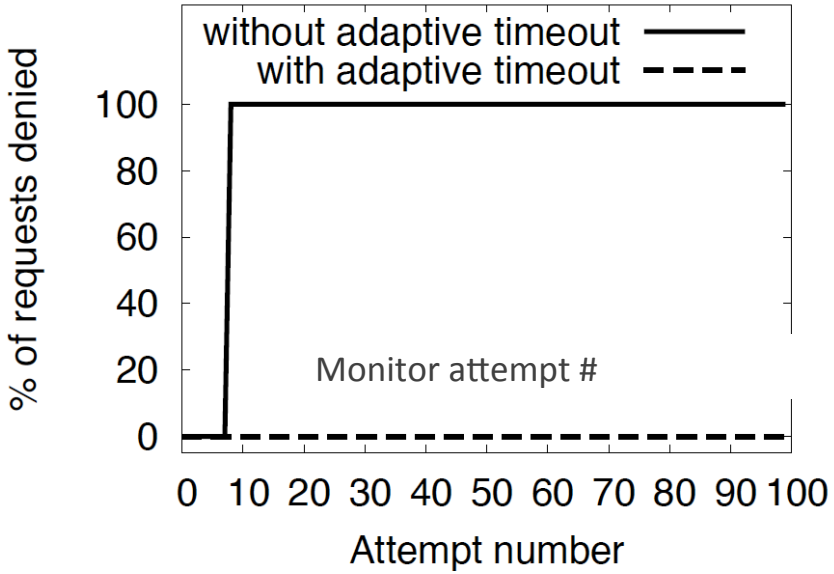
Adaptive Web Timeouts

- How timeouts should be setup?
  - Adaptive: when the server is busy, shorten the timeouts, and when the server is under-loaded increase timeouts
- Prototype implementation
  - Covers application, response and TCP timeouts
  - Implemented in Linux Kernel and Apache Web server
    1. The Web server communicates target rate to the kernel
    2. Connection speed is monitored by the kernel: Every second, a connection score is increased by the amount of transferred data, and reduced by the target rate
    3. If the send queue becomes empty, begin counting a new
    4. When 90% of connection slots are claimed, the Web server reduces application timeout to 3 seconds and signals to the kernel to terminate under-performing connections

Mechanism Demonstration

- Experiment:
  - Apache Web server with 256 concurrent connections
    - One time with adaptive response timeout that requires at least 500bytes/sec at times of stress
    - Another time without adaptive timeout
  - Attacker machine attempts to keep 300 connections open to the server and consumes responses at 200-300 bytes/sec
  - Monitor machine issues 100 requests in every attempt with 5s timeout (100 attempts,10 seconds apart), and observe the denial rate





Summary

- Measured different kinds of Web timeouts in the Internet
- Compared measured timeouts to normal client activity
  - Result: huge mismatch
- Proposed mechanisms for proper and secure provisioning of Web timeouts
  - Reduce timeouts only at times of high load

Questions?