TRODS

Transparent Recovery for Object Delivery Services

Wyatt Lloyd, Michael J. Freedman
Princeton University
Welcome

Message From the General Chair:

On behalf of the organizing committee, I warmly welcome you to join us in the 2011 International Conference on Dependable Systems and Networks, which will be held in Hong Kong, a vibrant city where East meets East, modern blends with tradition.

The Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) is the premier international conference for presenting the very best research results, problem solutions, and insight on new challenges facing the field of dependability and security. These challenges have always been significant in the IT-permeated modern society, and meeting them is essential to economic prosperity, global stability, and societal/personal safety. DSN has pioneered the fusion between security and dependability, addressing the need to simultaneously fight against cyber attacks, accidental faults, design errors, and unexpected operating conditions. This conference has been held annually since 1971 (the original conference name is IEEE International Symposium on Fault Tolerant Computing). Over the past 40 years, DSN has developed into the flagship conference in the systems research area, specializing in both dependability and security issues and solutions. The 41st DSN, to be held during June 27-30, 2011, will feature a rich program to host plenary talks, regular papers, experience reports, panels, student
Connection Recovered!
Object Delivery Services

• Read-Only

• Static Content

• Webpages, Images, Videos
Work Now

- Can’t Modify Clients
Key Idea

• Coerce client to help
  – To identify connections that need recovery
  – To reliably store information

• Yet client is **unmodified** and unaware
  – Exploit TCP spec to control client’s stack
Object Delivery Cluster

Service

Load Balancer

Server

Server

Server

Liveness Monitor
Failure

Service

Load Balancer

Server

Server

Server

Liveness Monitor

10
TRODS

Service

Client

Load Balancer

Server

Server

Server

Liveness Monitor
TRODS

Service

Client

Load Balancer

Server

Server

Server

Liveness Monitor

Store
# Road to Recovery

<table>
<thead>
<tr>
<th>Step</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redirect to live server</td>
<td>Liveness monitor updates load balancer</td>
</tr>
<tr>
<td>Induce client to send packet</td>
<td>Coerce client’s TCP stack</td>
</tr>
<tr>
<td>Continue Connection</td>
<td></td>
</tr>
<tr>
<td>Determine Phase</td>
<td>Use packet + stored info</td>
</tr>
<tr>
<td>Identify Object</td>
<td>Stored Info</td>
</tr>
<tr>
<td>Find Offset</td>
<td>Use packet + stored info</td>
</tr>
</tbody>
</table>
Coercing Clients

- Always Leave A Packet Unacknowledged

Exploit TCP Spec for Recovery Initiation!

Always Something Here
Continuing the Connection

• Determine Phase:
  1) TCP Setup
  2) HTTP Setup
  3) HTTP Download
  4) TCP Teardown

TRODS Saves Info
Continuing the Download

- HTTP ObjectID

- Offset = TCP Ack – HTTP ObjectISN
Continuing the Download

- **HTTP ObjectID**

- **Offset = TCP Ack − HTTP ObjectISN**
Persistent Store

• Key-Value Store
  + Corner Cases Handled
  + Unlimited Objects
  – Still Efficient (1 save only)

• TCP Timestamp
  + Very Efficient (1 machine only)
  – 1 Million Object Limit
  – Corner Cases

Exploit TCP Spec for Persistence!
Recover the Connection

• Initiate New Connection
  – GET ObjectID ...
  – Range: bytes=Offset-

• Splice Connections Together

• Works with Unmodified Servers!
TRODS

1) Packet Manipulation
TRODS

1) Packet Manipulation
2) Protocol Inspection
TRODS

1) Packet Manipulation
2) Protocol Inspection
3) Blocks Connection
TRODS

1) Packet Manipulation
2) Protocol Inspection
3) Blocks Connection
4) State Injection
TRODS

1) Packet Manipulation
2) Protocol Inspection
3) Blocks Connection
4) State Injection
5) Recovery Initiation

Server

TCP
TRODS
IP

Ack
Failure Walkthrough

Service

Liveness Monitor

KV Store

Load Balancer

Server

Server

Response

ID

ISN

Request
Failure Walkthrough

Service

Load Balancer

Server

Liveness Monitor

KV Store

ACK

FIN

?
Related Work

• New Transport
  – Trickles, SCTP, TCP Migrate, …

• TCP
  – FT-TCP, ST-TCP, Backdoors, …

• HTTP
  – CoRAL, …
Implementation

• Linux Kernel Module

• 3,000 lines of C

• ~CoRAL
  – Optimistic subset of CoRAL
Experiments

• Additional Latency
  – Normal
  – Failure

• Throughput
  – Lighttpd @ Princeton
  – Apache @ Emulab
  – Hybrid TS & KV Throughput
  – Failure
Normal Case Latency

- **TRODS-TimeStamp (TS)**
  - Median: + 0.009 ms
  - 99\(^{th}\) : + 0.012 ms

- **TRODS-Key-Value (KV)**
  - Median: + 0.137 ms
  - 99\(^{th}\) : + 0.148 ms
Recovery Latency

CDF

Additional Latency

0

~0

.2ms

20ms

200ms

3s

Blink of an eye

~15%

~35%

~50%
ThroughPut Per Server

120 ops/s

Raw

120 ops/s

30 ops/s/server

Frontend

30 ops/s/server

TPPS  20 ops/s/server
Apache

Normalized TPPS vs. Web Object Size

- Unmodified
- TRODS-TS
- TRODS-KV
- FT-TCP(cold)
- FT-TCP(hot)
- CoRAL
Summary

• Recover Object Delivery Connections

• Exploit TCP Specification to Coerce Clients
  – To send recovery-starting packets
  – To provide persistent storage

• Evaluation
  – Low Latency
  – High Throughput Per Server
Summary

• Recover Object Delivery Connections

• Exploit TCP Specification to Coerce \( ^\wedge \text{Clients} \)
  – To send recovery-starting packets
  – To provide persistent storage

• Evaluation
  – Low Latency
  – High Throughput Per Server

• Questions?