# Timing Attacks Made Practical

**Timothy D. Morgan**
Blindspot Security

**Jason W. Morgan**
Ohio State

black hat
USA 2015

**Timothy D. Morgan**

Founder & Chief Pwner

Blindspot Security

**Jason W. Morgan, Ph.D.**

Post-Doctoral Researcher

The Ohio State University

# Background

# Timing Side-Channel Attacks

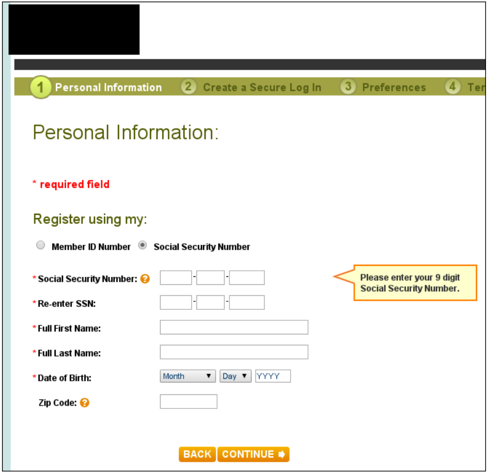
- Security-critical decisions
- Returns result to user, but *how* it decides is secret
- Computation time exposes decision details

# Examples of Timing Attacks

- Numerous crypto examples:
  - Cache-Timing Attacks on AES – DJB, 2005
  - Cache Missing for Fun and Profit. – Percival, 2005
  - Lucky Thirteen – AlFardan et. al., 2013
- What about web apps?

# Web Application Timing and KBA

Knowledge-Based Authentication could be ripe for abuse



[redacted]

1 Personal Information 2 Create a Secure Log In 3 Preferences 4 Ter

## Personal Information:

* required field

Register using my:

○ Member ID Number ● Social Security Number

* Social Security Number: ? [ ]-[ ]-[ ]

Please enter your 9 digit Social Security Number.

* Re-enter SSN: [ ]-[ ]-[ ]

* Full First Name:

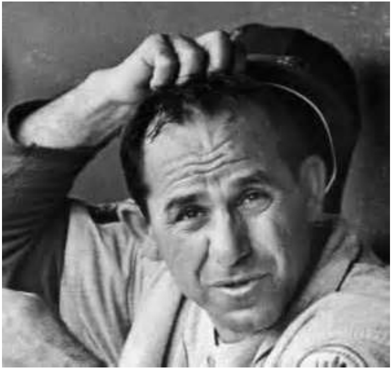
* Full Last Name:

* Date of Birth: Month ▼ Day ▼ YYYY

Zip Code: ?

BACK CONTINUE

# Motivation



*In theory there is no difference between theory and practice.*
*In practice there is.* – Yogi Berra

# Theory vs. Practice

- Most past research is:
  - Limited to specific vulnerabilities
  - Only tested under synthetic network conditions
- Very few tools available (namely Time Trial)
- Lack of thorough statistical analysis to establish scope conditions

# Goals

- Improve on statistical methods
- Be able to answer the question:
  "is this timing flaw I just found practically exploitable?"
- Investigate TCP Timestamps

# Data Collection

# Paired Sampling

- Two or more “test cases” are defined
- Each “sample” is a tuple of probes
- Probes in a sample are collected at the same time

# What are TCP Timestamps?

- Added to TCP to improve efficiency
- A host timestamp added to every header
- FMI: RFC 1323

# Getting at TCP Timestamps

- A sniffer is basically required
- TSval clock frequency estimation is also tricky
- Down-side: Complex packet analysis
- Up-side: More accurate RTT measurement

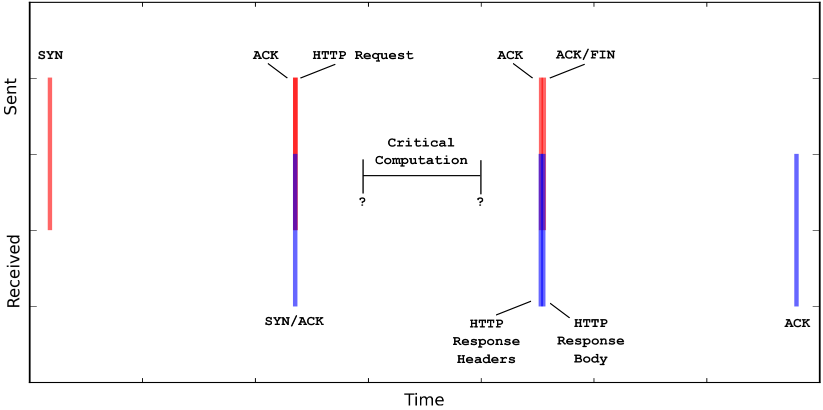
# TSval Precision Issues

- No specific clock frequency/precision required by RFC
- Different OSes/hardware use different frequencies
- Starting point for TSvals can be different for each TCP connection
- Typically tied to a RTC (with skew)

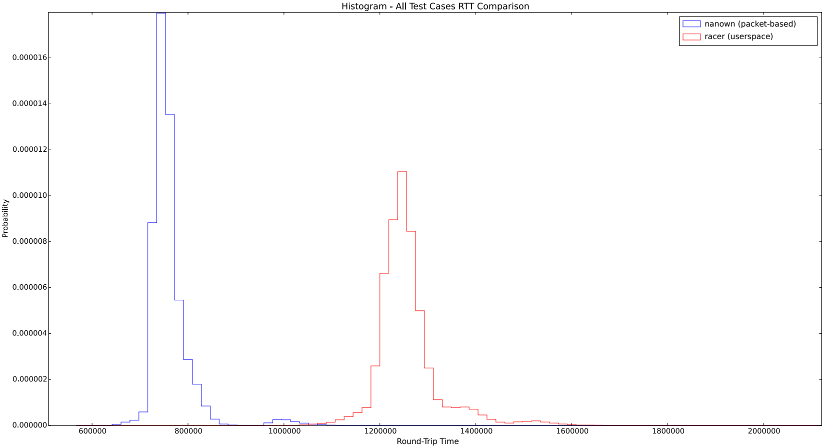
# TSval Precision Estimation

- Trickle HTTP request slowly to host (this forces many ACK responses)
- Sniff TSvals, apply least-squares regression
- Wash, rinse, repeat. Average results

# A Simple HTTP Request



# Packet Sniffing Yields RTT Measurement Bonus



# Statistical Analysis

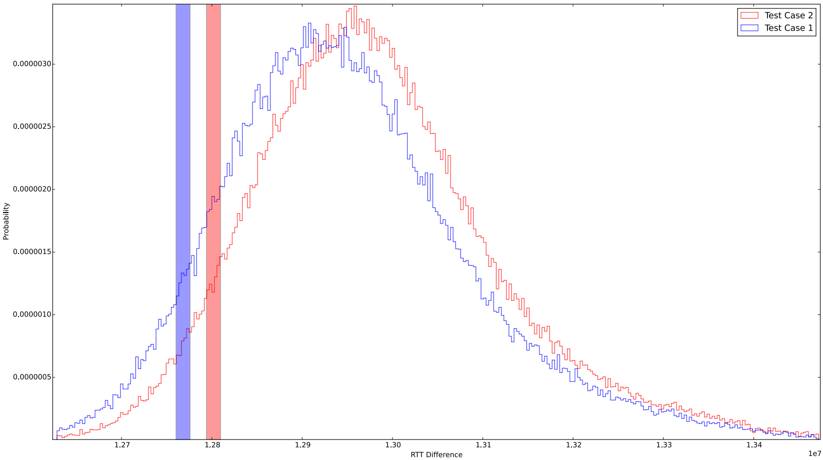
# Robust Statistics Required

- Network data is really noisy
- Basic measures, such as the mean, break down quickly
- “Robust statistics” or ways to filter noise are needed

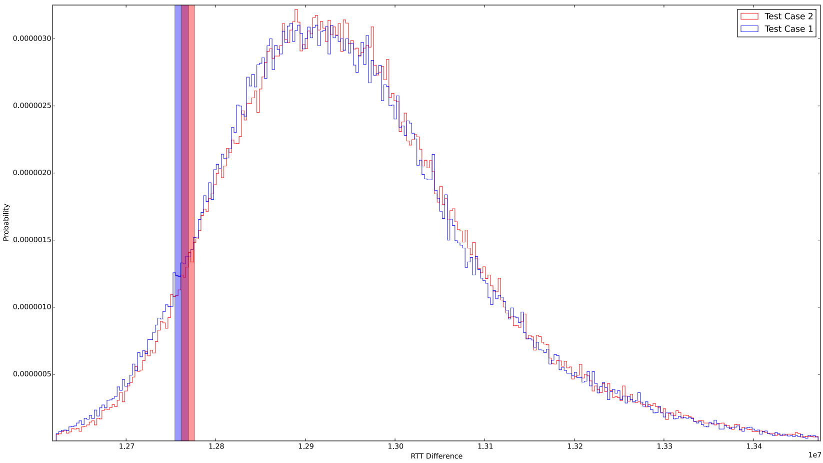
# The Venerable Box Test

- A type of $L$-statistic apparently pioneered by Crosby, et.al.
- Two parameters: “low” and “high” percentiles define the “box”
- Compare two distributions to see if boxes overlap

# Box Test - Classified as Different



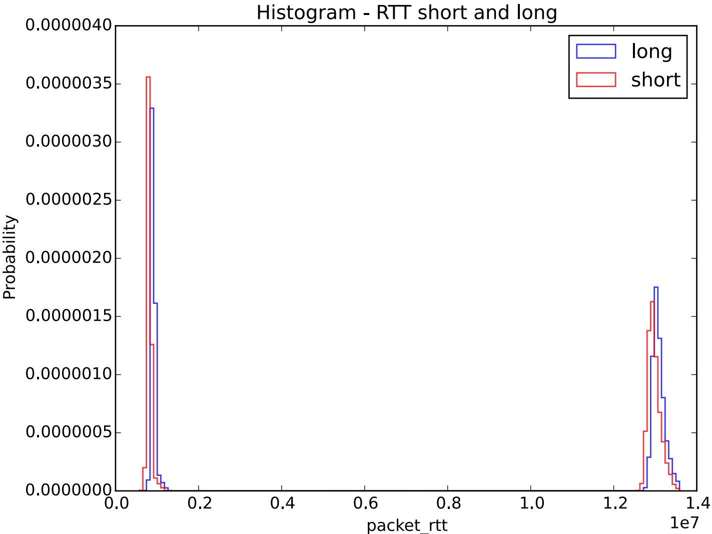
# Box Test - Classified as the Same



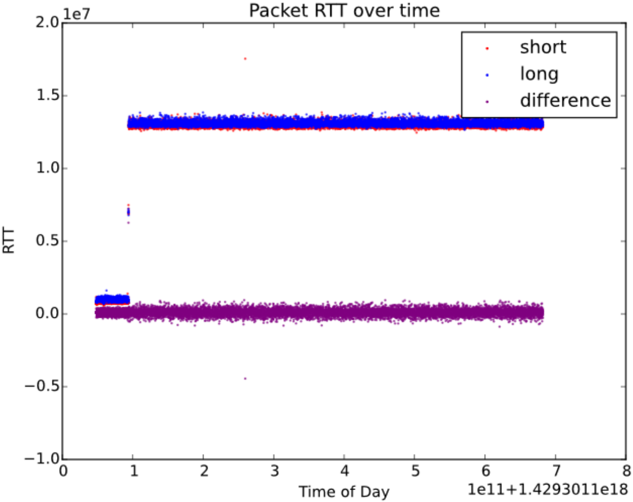
# Box Test - Training

- No official training algorithm
- We train 2 parameters: box location and width
- 4-step iterative algorithm to avoid $O(N^2)$
- Bootstrap and measure error rates at each stage

# Problem with Independent Distributions



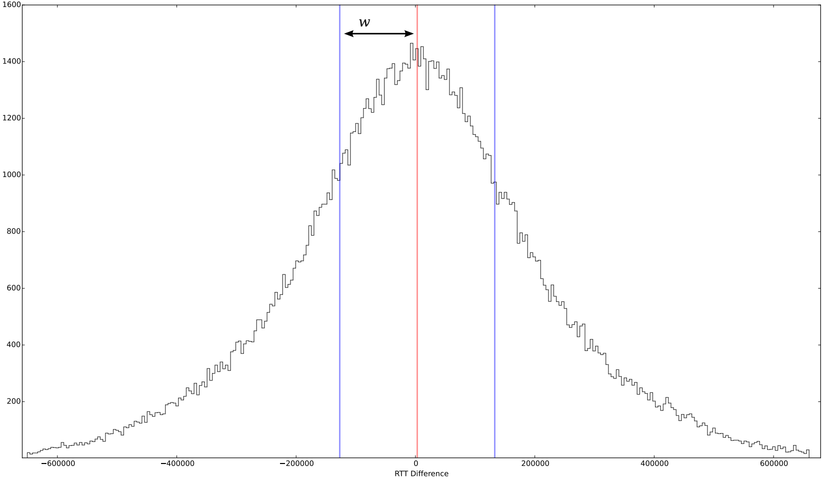
# Why Not Use the Distribution Pair-Wise of Differences?



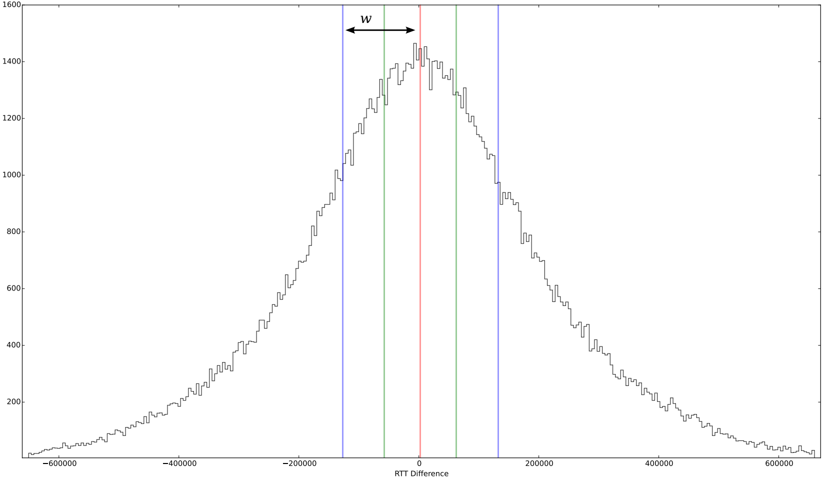
## $L$-Estimators

- Order statistics: the median, the 37th percentile, midhinge, ...
- $L$-estimators: linear combinations of order statistics
- Very simple to calculate and robust, but not “efficient” in a statistical sense

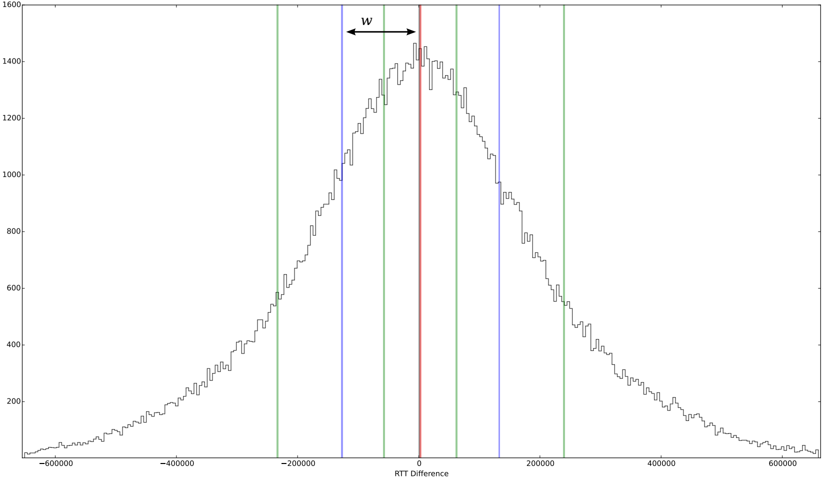
# midsummary


w
1600
1400
1200
1000
800
600
400
200
−600000
−400000
−200000
0
200000
400000
600000
RTT Difference

# quadsummary



# septasummary



# $L$-estimator Training

- Train two parameters: $w$ and threshold
- Threshold starts at $1/2$ the estimate
- 4-step bootstrap similar to box test's

# TCP TSval Mean

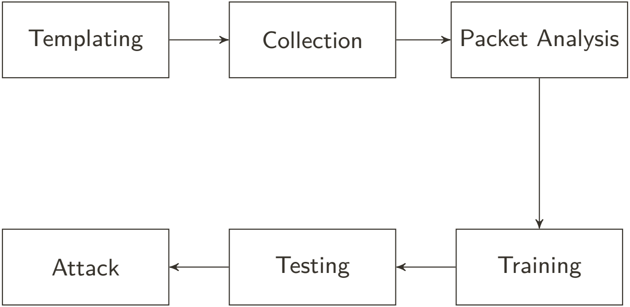
- If your watch ticks once per second, can you measure a 1ms event?
- Yes, if you can gather lots of samples
- Out of 10000 samples, how many should have a 1sec reading?
- No luck with this yet though :-(

# A Tool: *nanown*

# Nanown

- Identify timing leaks
- Quantify risk
- Exploit
- As with all open source, a work in progress...

# Nanown Work-flow



# Nanown Train/Test Process

- Trains all classifiers on $\sim$19 sample sizes
- Tests each candidate parameters
- Zeros in on minimum sample size needed for 95% confidence

# Monte Carlo Analysis

# Test Scenarios

Table : Network Scenarios

| Name | Type | OS | Network<br>Hops | Approx.<br>Latency (ms) | TSval<br>Precision (ms) |
|---|---|---|---|---|---|
| lnx | physical | Linux 3.16 | 1 | 0.25 | 4.00 |
| vm | Qemu VM | Linux 3.16 | 2 | 12.00 | 4.00 |
| vps | Linode VM | Linux 4.0 | 12 | 31.00 | 3.33 |
| bsd | physical | FreeBSD 10.1 | 13 | 84.00 | 1.00 |

# Sampling

- 5 Timings each (except one scenario):
  40ns, 200ns, 1000ns, 5000ns, 25000ns
- Samples: 250,000 each (500,000 individual probes)
- Separate train & test data
- 1000 iterations for each observation size in final test runs

# Results

Table : Number observations if $< 5\%$ error; percent error otherwise

| | | Delta (ns) | | | | |
|---|---|---|---|---|---|---|
| | Classifier | 25000 | 5000 | 1000 | 200 | 40 |
| lnx | | | | | | |
| | midsummary | 29 obs | **894 obs** | 17147 obs | **16.60% err** | **38.60% err** |
| | quadsummary | 26 obs | **894 obs** | **16289 obs** | 20.55% err | 47.30% err |
| | septasummary | **15 obs** | **894 obs** | 17147 obs | 22.35% err | 45.20% err |
| | boxtest | 146 obs | 20.80% err | 36.30% err | 47.55% err | 49.85% err |
| vm | | | | | | |
| | midsummary | **242 obs** | 10898 obs | **15789 obs** | 19.45% err | **23.05% err** |
| | quadsummary | 344 obs | 10583 obs | 8.30% err | **18.40% err** | 30.05% err |
| | septasummary | 356 obs | 9706 obs | 8.30% err | 22.40% err | 31.10% err |
| | boxtest | 615 obs | **7909 obs** | 7.50% err | 47.00% err | 36.00% err |
| vps | | | | | | |
| | midsummary | **21.80% err** | 31.80% err | **19.00% err** | 33.10% err | **35.85% err** |
| | quadsummary | 32.75% err | **31.55% err** | 34.95% err | **32.25% err** | 37.35% err |
| | septasummary | 22.40% err | 43.50% err | 30.05% err | 46.55% err | 36.70% err |
| | boxtest | 48.15% err | 39.70% err | 41.00% err | 46.70% err | 44.75% err |
| bsd | | | | | | |
| | midsummary | **21.30% err** | 21.80% err | | | |
| | quadsummary | 22.35% err | 28.65% err | | | |
| | septasummary | 27.65% err | **18.00% err** | | | |
| | boxtest | 24.35% err | 46.80% err | | | |

# Demo

# Intentionally Vulnerable KBA

- Implemented KBA registration form
- Timing difference between most fields

# Conclusion

# Our Contributions

- Less noise through packet-based RTT collection
- More resilient classification method
- A tool that assists in risk evaluation and exploitation

# Avoidance

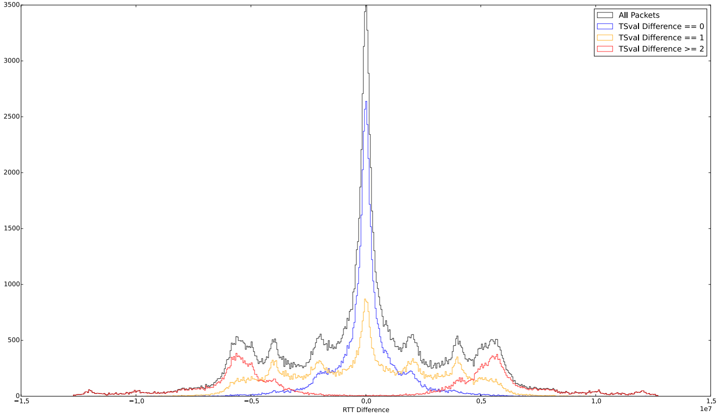
- Implement time-constant logic where possible
- Add CAPTCHAs to forms with user interaction
- Test for timing differences in critical operations

# Take Aways

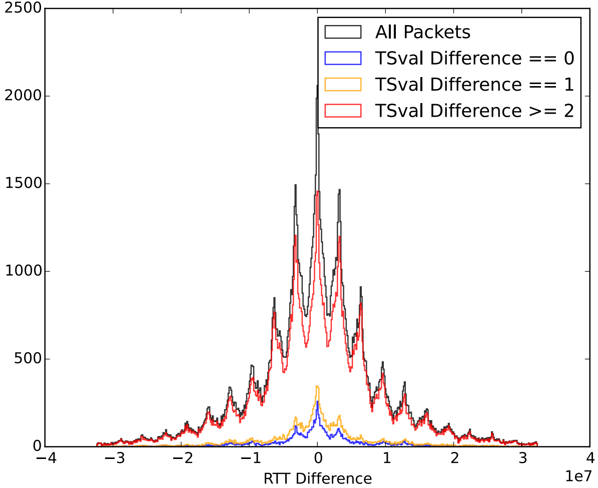
- Remote timing attack techniques are still in their infancy
- Except for string comparision, most timing differences are exploitable on the LAN
- Exploitation over the Internet is harder

# Questions?

# TCP Timestamps - Partitioning on lnx



# TCP Timestamps - Partitioning on bsd


1e7
RTT Difference
All Packets
TSval Difference == 0
TSval Difference == 1
TSval Difference >= 2