Tick Tock: Building Browser Red Pills from Timing Side Channels

Grant Ho, Dan Boneh and Lucas Ballard, Niels Provos
Stanford Google

Terminology: Red Pills

- **Red Pill:** code that distinguishes between a real execution environment vs. emulated/virtualized environment

- **Native Red Pills:** Red pills that execute at application layer (directly on OS).

- **Browser Red Pills:** Red pills executed in the browser sandbox (embedded in webpages)

Prior Work on Red Pills: Differentiating VMs from Real Machines

- Black Hat 2005: Original Native Red Pill by J. Rutkowska
- Since then, many other papers on native red pills (Chen 2008, Franklin 2008, Kapravelos 2011, Paleari 2009, Shi 2014, etc.)

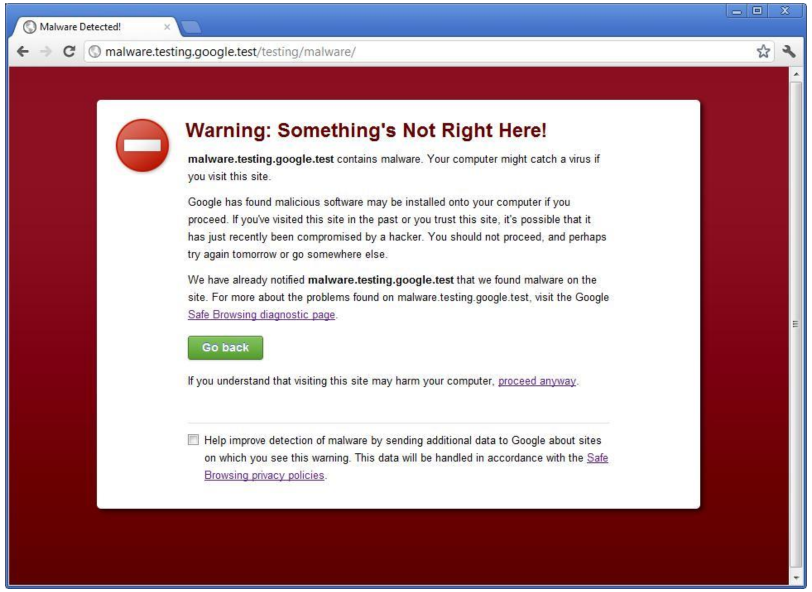
- **Key Problem:** Native red pills rely on low-level operations (examining registers, looking for debugging processes, etc.)
  - Inaccessible to websites (Javascript in the browser)

- **This Talk:** Designing browser red pills

Motivation

- **Perspective:** Malicious website (e.g. drive-by download)

- **Challenge:** Anti-malware honeypots (i.e. instrumented VMs)



Warning: Something's Not Right Here!

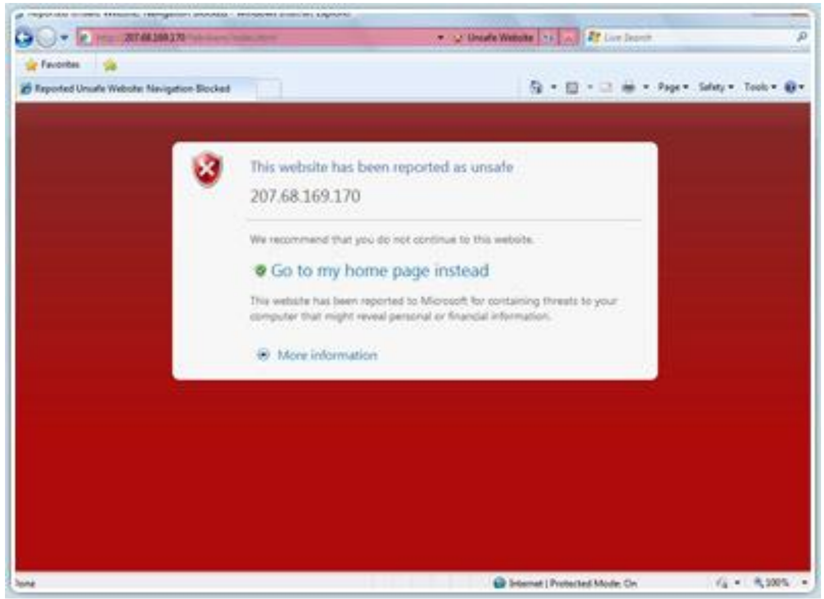
malware.testing.google.test contains malware. Your computer might catch a virus if you visit this site.

Google has found malicious software may be installed onto your computer if you proceed. If you've visited this site in the past or you trust this site, it's possible that it has just recently been compromised by a hacker. You should not proceed, and perhaps try again tomorrow or go somewhere else.

We have already notified **malware.testing.google.test** that we found malware on the site. For more about the problems found on malware.testing.google.test, visit the Google Safe Browsing diagnostic page.

If you understand that visiting this site may harm your computer, proceed anyway.

□ Help improve detection of malware by sending additional data to Google about sites on which you see this warning. This data will be handled in accordance with the Safe Browsing privacy policies.



207.68.169.170
Unsafe Website
Live Search
Favorites
Reported Unsafe Website: Navigation Blocked
Page
Safety
Tools
Internet | Protected Mode: On
100%

This website has been reported as unsafe

207.68.169.170

We recommend that you do not continue to this website.

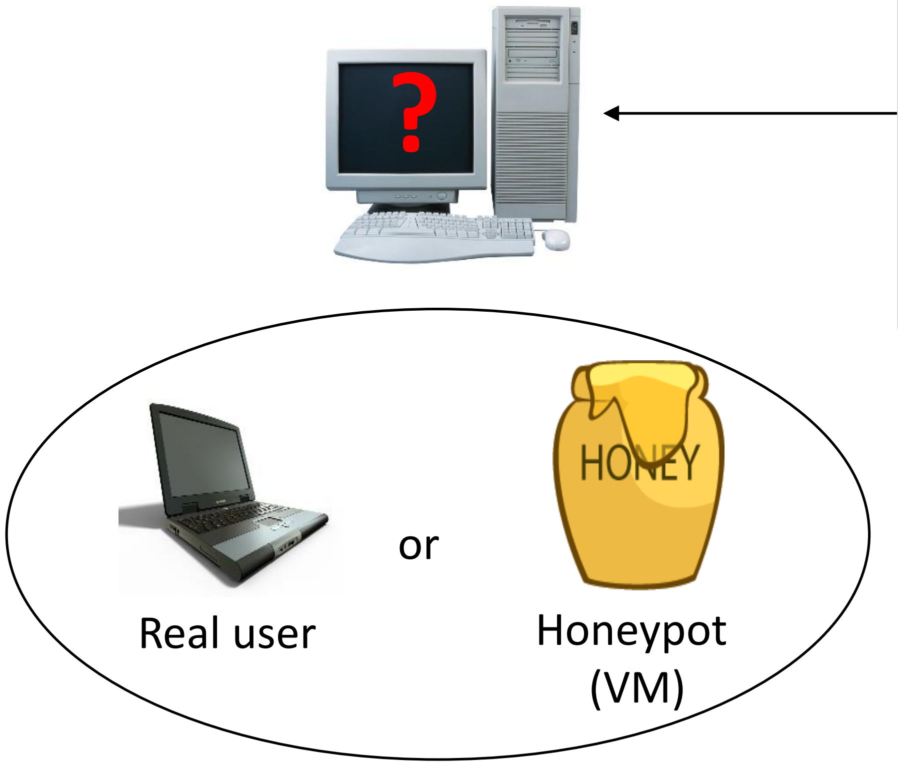
Go to my home page instead

This website has been reported to Microsoft for containing threats to your computer that might reveal personal or financial information.

More information

- **Goal:** deliver malicious web content to normal users, but act benignly in anti-malware honeypots (VMs).

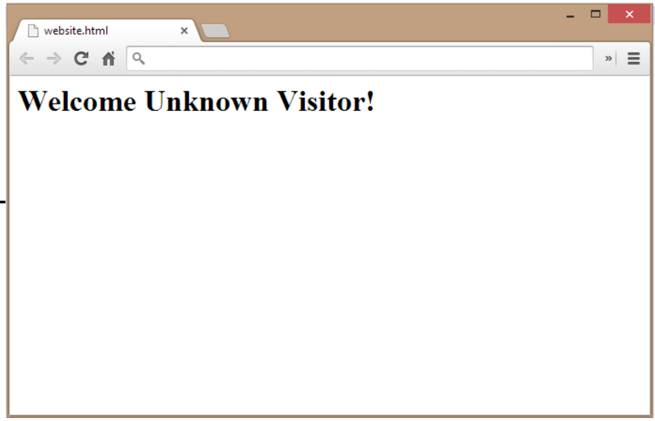
Attack Model



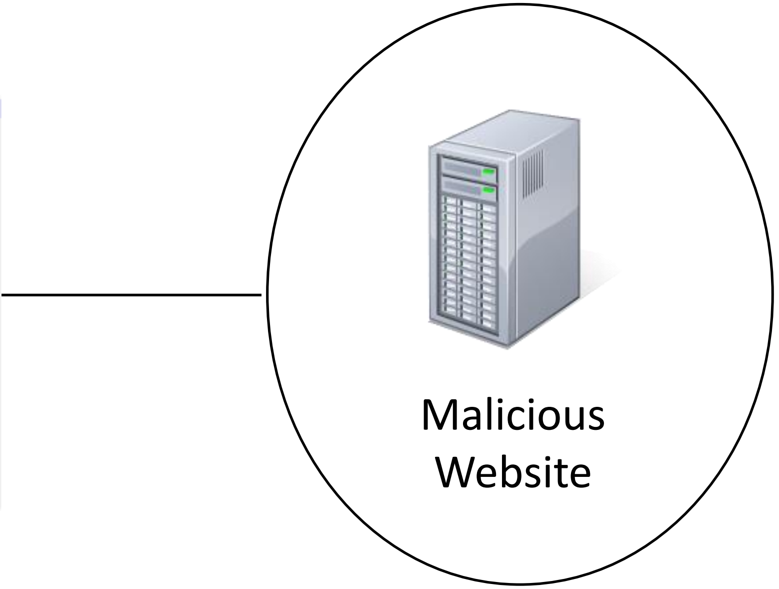
Real user

or

Honeypot
(VM)

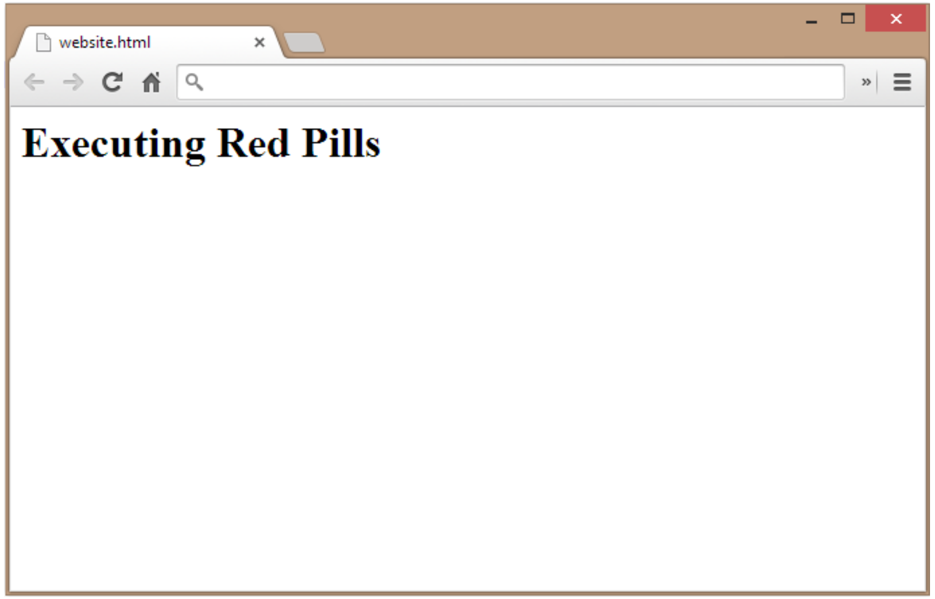


Welcome Unknown Visitor!



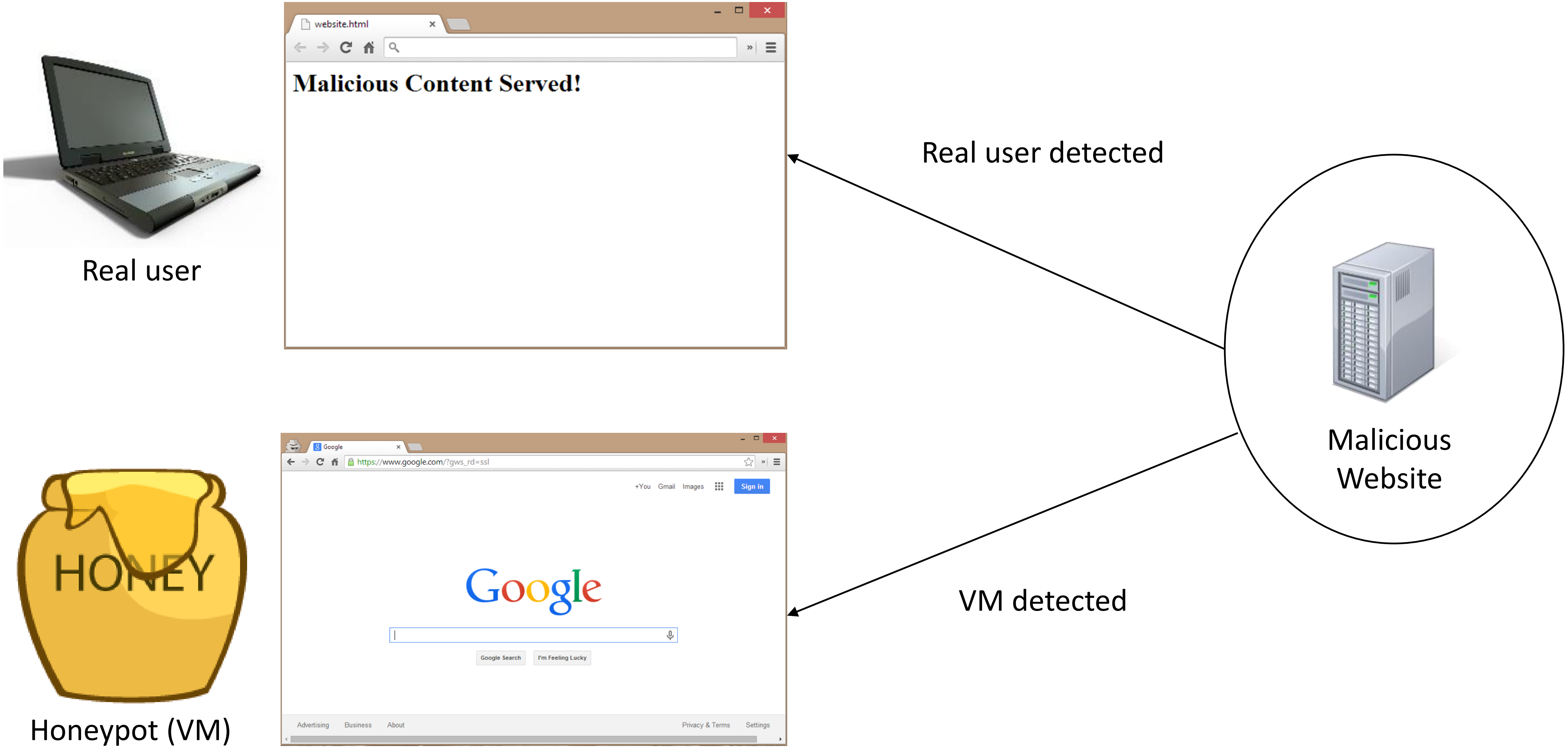
Malicious
Website

Attack Model



Executing Red Pills

Attack Model



website.html

Malicious Content Served!

Real user

Real user detected

Malicious Website

VM detected

Honeypot (VM)

Design and Implementation of Browser Red Pills

Distinguishing VM vs. non-VM from the Browser

- Can’t access low-level configuration anomalies, but what about performance anomalies?

- Side effects of a virtualization = altered behavior/performance of certain operations?

- **Main Idea:** Measure execution time of certain Javascript operations (***differential operations***) to detect virtualization performance anomalies

Constructing Timing-Based Red Pills

- **Problem:** How do you account for performance differences that results from different hardware or background activity?
  - Raw execution times unreliable

- **Solution:** Use relative timing measurements for red pills

Constructing Timing-Based Red Pills

- Idea: Run a simple Javascript operation (***baseline operation***) to gauge background load/base performance
    - Also, baseline operation should not have performance difference between VM and non-VM

- Adjust red pill’s timing measurement for background load by: dividing the differential operation’s execution time by this baseline time

- Timing done in Javascript (granularity: 1 millisecond)

Browser Red Pills Schematic

- Three steps to browser red pill, which returns a number (timing ratio).

Browser Red Pill

Step 1: Execute baseline op and measure execution time

Step 2: Execute differential op and measure execution time

Return:
differential_op_time / baseline_op_time

Differential Operations

1. Writing to Console
2. Local Storage
3. Spawning Web Workers (Javascript Threads)
4. Communicating between Web Workers
5. Communicating between CPU and GPU via WebGL
6. Heavy Graphics Rendering via WebGL

Baseline Operations

1. Repeatedly writing lots of text to a DOM node
2. Allocating large chunks of memory

Evaluation and Results

Testing Environment

- Windows 7 and Windows 8.1 on real machines, VMWare in binary translation mode, and VMWare in hardware-assisted virtualization mode (6 system environments)

- Chrome 34, Firefox 29, IE 11 (3 browser environments)

- 18 Total environments: {Chrome, Firefox, IE} x {Real, VM-BT, VM-HV} x {Win 7, 8}

- Loaded each red pill webpage 100 times w/ 500 ms in-between reloads

Evaluation Methodology

- Now we have distributions (100 measurements) of red pill timing ratios

- For each red pill on a given {browser, OS} combination:
  - Compare distribution of timing ratios for: Real Machine vs. VM-BT
  - Compare distribution of timing ratios for: Real Machine vs. VM-HV

- **Key question:** Is there a significant and practical difference between a red pill’s timing ratios on a VM vs. non-VM?

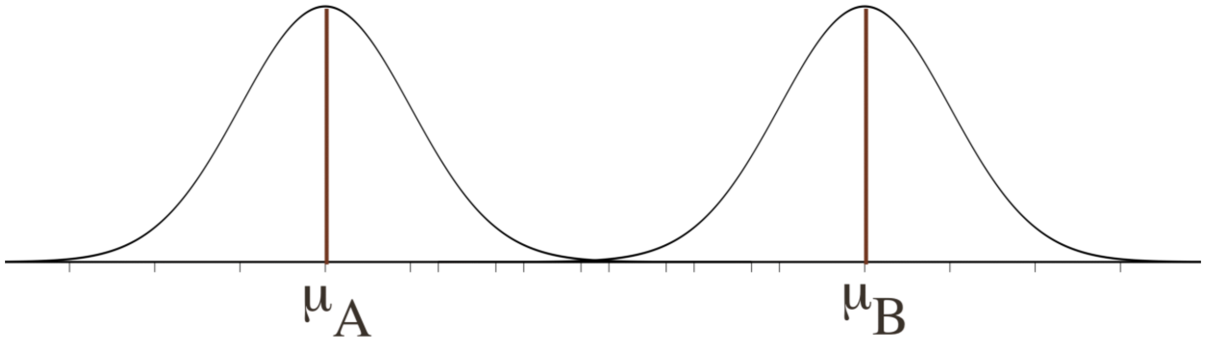
Evaluation Metrics

- Two criteria (both should be satisfied):

1. Does a t-test of VM timing ratios vs. non-VM timing ratios produce a p-value less than 0.05? (statistically significant difference)
2. Is there a "small" overlap between distribution of timing ratios for VM vs. non-VM? (practical difference)

Criteria 1: Independent, Two-Sample T-Test

- “Welch’s t-test”: two independent samples with unknown population variances



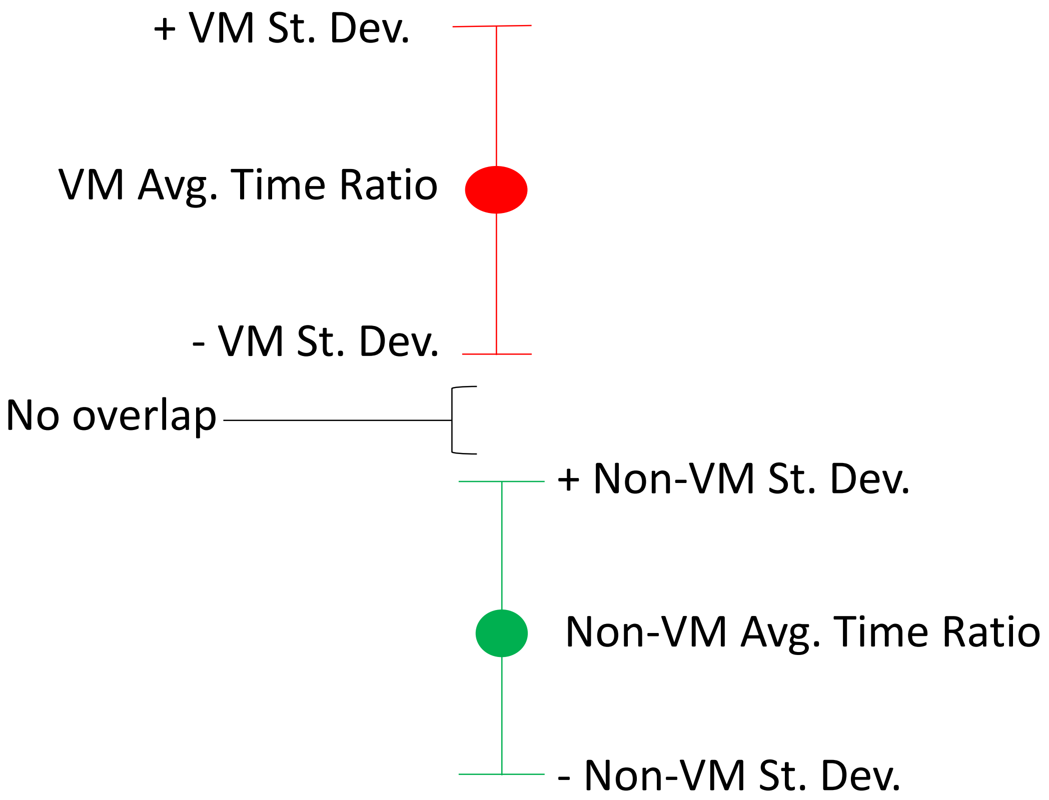
Is there a statistically significant difference between *distribution A* (e.g. real machine timing ratios) and *distribution B* (e.g. VM timing ratios)?

Criteria 1: Two-Sample T-Test (Cont.)

- ***p-value*** from t-test: probability that there is no statistical difference between the two distributions
- Criteria 1: t-test of average timing ratios of VM vs. non-VM has

  p-value < 0.05 (all our successful red pills have p-value < 0.000001)

Criteria 2: Minimally Overlapping Distributions



- Magic-number cut-off: high probability of evading of VMs and targeting of real machines

Results: Number of Successful Red Pills per Environment

	Chrome	**IE**	**Firefox**
Windows 8 BT	6	3	4
Windows 8 HV	6	3	4
Windows 7 BT	3	5	4
Windows 7 HV	3	5	4

- BT = Binary Translation VM, HV = Hardware-assisted virtualization VM
- Numbers represent sum of successful red pills from both DOM baseline + Memory baseline

Defenses for Honeypots

Distorting Time in the Honeypot

- Replace Javascript timing functions with fake timing
  - Return random or faster timing measurements

- Replace Javascript/HTML5 operations with fake functions
  - Cheat on differential operation execution by completing early/faking execution

- These are weaker defenses… likely induce other anomalies.

Detecting Javascript Evasion (Red Pills)

- Detect differential operations
  - Efficacy unclear for many operations (e.g. Console writing and Graphics)

- Detect baseline operations
  - More practical and effective against current baseline operations

- Compare rendered content in non-VM and VM to find evasive JS (Kapravelos 2013, Kirat 2014)

Conclusion

Summary

- Browser red pills are possible via timing side-channels
  - Timing/performance analysis overcome restricted computing environment
  - Baseline operations resolve background load/noise

- Highly effective:
  - Three major browsers on two latest versions of Windows
  - Hardware-assisted virtualization and binary translation

- Detecting baseline operation most promising defense

- Future work: (1) Measure prevalence of existing JS red pills
  (2) Understand JS implementation diffs in good red pills

Questions?