UNIVERSITY OF OREGON

Leveraging USB to Establish Host Identity

Adam Bates Ryan Leonard Hanna Pruse

Kevin Butler Daniel Lowd

OSIRIS Oregon Systems Infrastructure Research & Information Security Laboratory

NDSS'14, San Diego, CA, USA
25 February, 2014

Dude, ARE you getting a Dell?

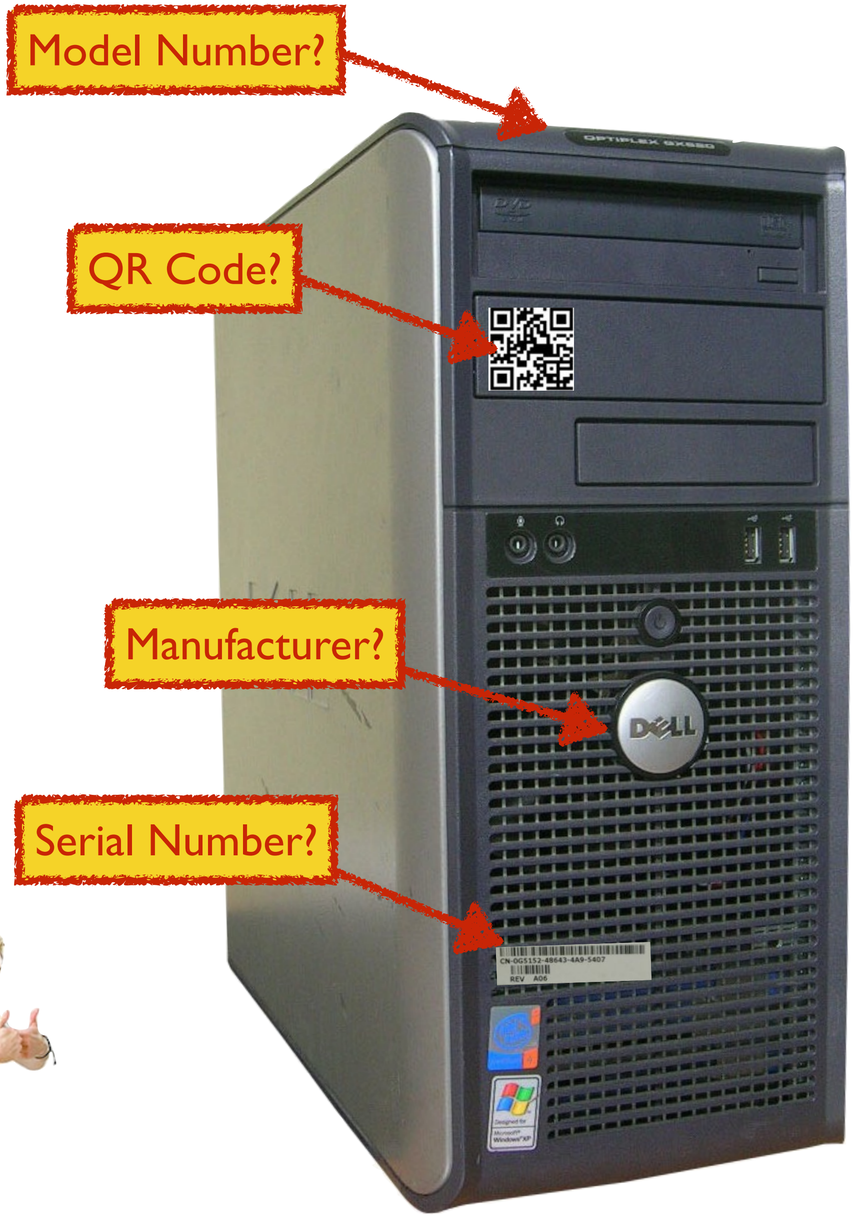
- Establishing host identity is surprisingly hard.
- How can we confidently answer the following questions?
  - *Was this machine replaced with one under adversary-control?*
  - *Has hardware in this machine been replaced by an adversary?*
  - *Is this hardware-based attestation vulnerable to relay attacks?*

Dude, ARE you getting a Dell?

- Establishing host identity is surprisingly hard.
- How can we confidently answer the following questions?
  - *Was this machine replaced with one under adversary-control?*
    (Manufacturer, Model Number)
  - *Has hardware in this machine been replaced by an adversary?*
  - *Is this hardware-based attestation vulnerable to relay attacks?*

Dude, ARE you getting a Dell?

- Establishing host identity is surprisingly hard.
- How can we confidently answer the following questions?
  - *Was this machine replaced with one under adversary-control?*
    (Manufacturer, Model Number)
  - *Has hardware in this machine been replaced by an adversary?*
    (Serial Number, QR Code)
  - *Is this hardware-based attestation vulnerable to relay attacks?*



Model Number?

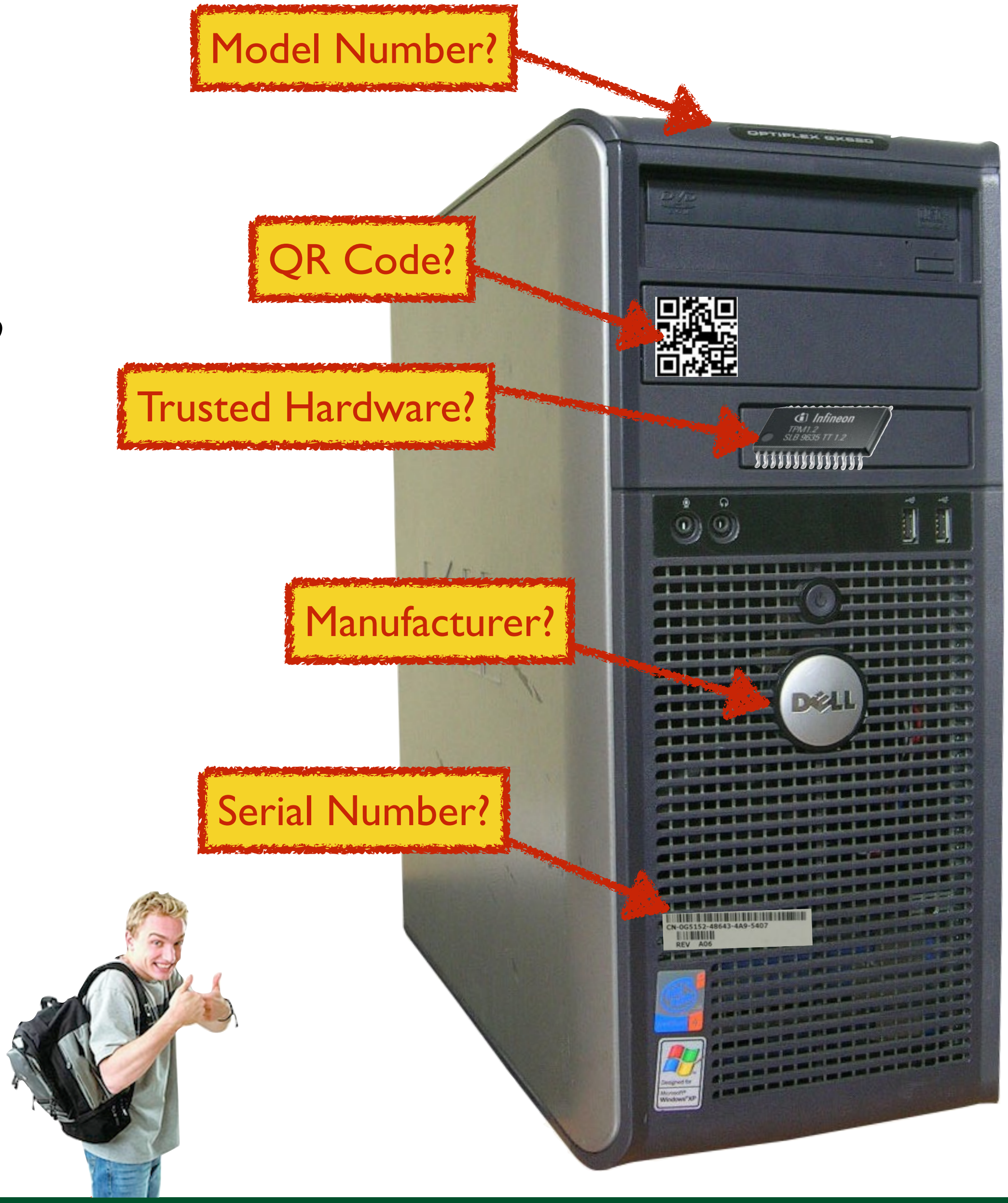
QR Code?

Manufacturer?

Serial Number?

Dude, ARE you getting a Dell?

- Establishing host identity is surprisingly hard.
- How can we confidently answer the following questions?
  - *Was this machine replaced with one under adversary-control?*
    (Manufacturer, Model Number)
  - *Has hardware in this machine been replaced by an adversary?*
    (Serial Number, QR Code)
  - *Is this hardware-based attestation vulnerable to relay attacks?*
    (Trusted Hardware, e.g., TPM)



Model Number?

QR Code?

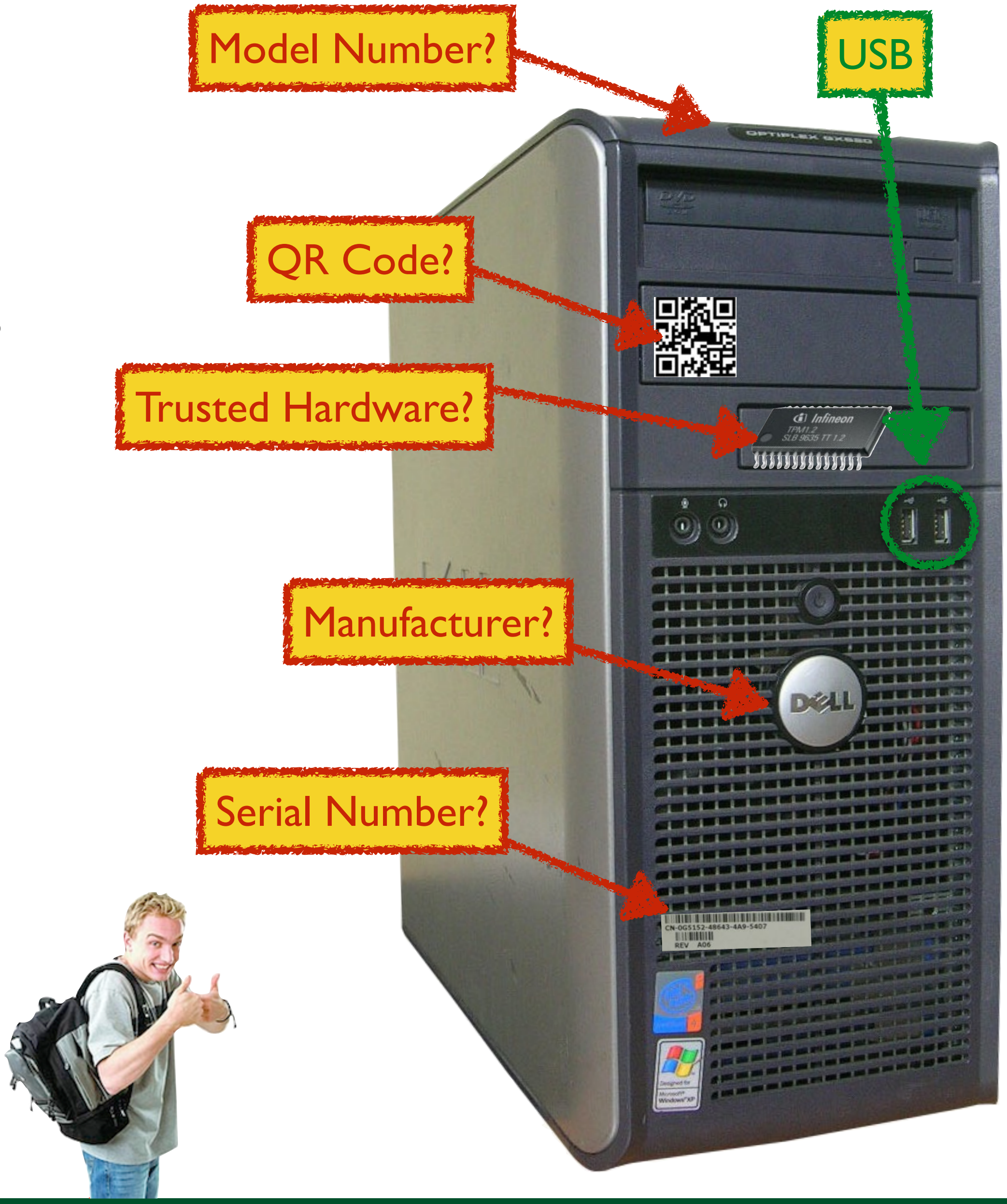
Trusted Hardware?

Manufacturer?

Serial Number?

Dude, ARE you getting a Dell?

- Establishing host identity is surprisingly hard.
- How can we confidently answer the following questions?
  - *Was this machine replaced with one under adversary-control?*
    (Manufacturer, Model Number)
  - *Has hardware in this machine been replaced by an adversary?*
    (Serial Number, QR Code)
  - *Is this hardware-based attestation vulnerable to relay attacks?*
    (Trusted Hardware, e.g., TPM)



Model Number?

USB

QR Code?

Trusted Hardware?

Manufacturer?

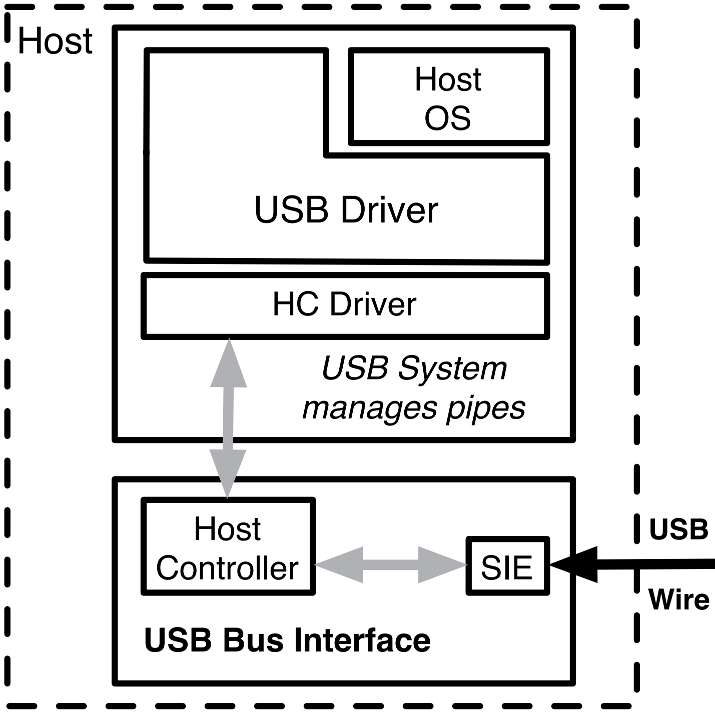
Serial Number?

Can USB bootstrap trust?

- Our work demonstrates that USB interactions can be observed to not only infer the attributes of a host, *but to also to build a uniquely identifiable machine fingerprint.*
- Using machine learning techniques, we show that USB timing information can be used to identify…
    - … host attributes (e.g. OS, Model) with over 90% accuracy.
    - … 70% of a field of 30 seemingly identical machines!
- We develop an Android App that turns your smart phone into a USB Protocol Analyzer.

Background: USB

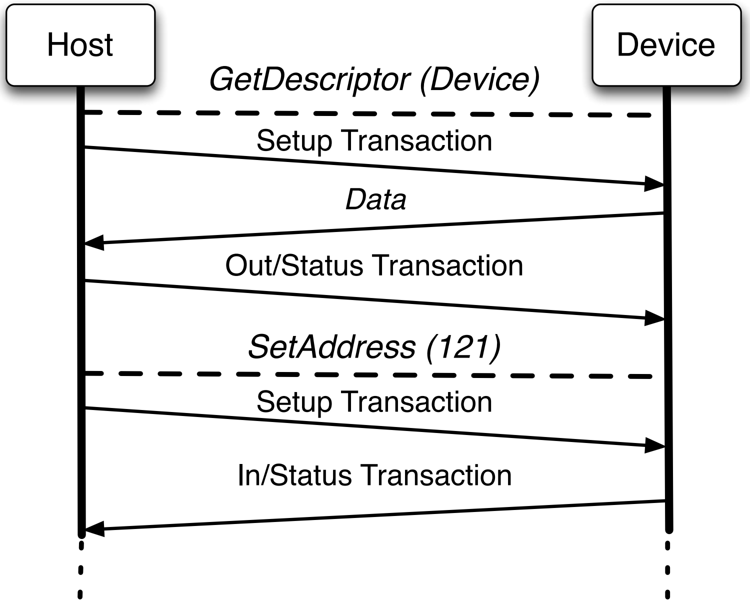
- USB behavior is impacted by Host OS, USB Driver, Firmware, USB Board, and more.
- We observe just the *enumeration* phase in which the host discovers and configures the device.
- Enumeration can be forced on any powered-on USB board that is configured to run in host mode.



Overview of the Host USB Stack.

Background: USB

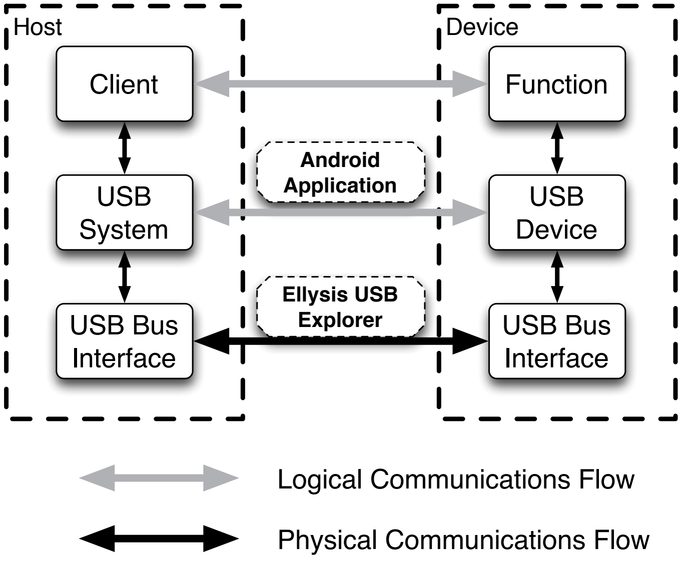
- USB behavior is impacted by Host OS, USB Driver, Firmware, USB Board, and more.
- We observe just the *enumeration* phase in which the host discovers and configures the device.
- Enumeration can be forced on any powered-on USB board that is configured to run in host mode.



An example USB data flow. Control transfers (dotted lines) are comprised by a set of transactions (solid lines)

Android USB Analyzer

- Developers use protocol analyzers to debug USB activity as it physically traverses the wire.
- Our Android app for rooted phones loads a modified USB Driver with *kprobe*.



Host

Client

USB System

USB Bus Interface

Android Application

Ellysis USB Explorer

Device

Function

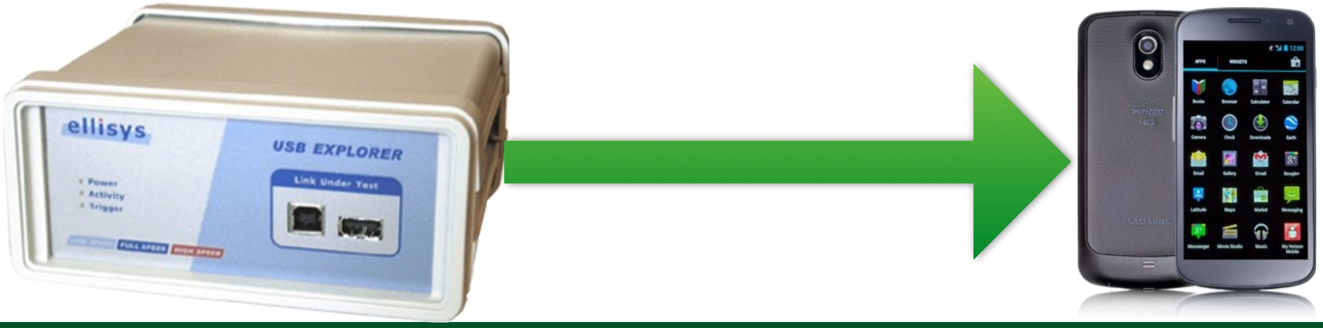
USB Device

USB Bus Interface

Logical Communications Flow

Physical Communications Flow

Our Android Protocol Analyzer observes logical USB communications flows

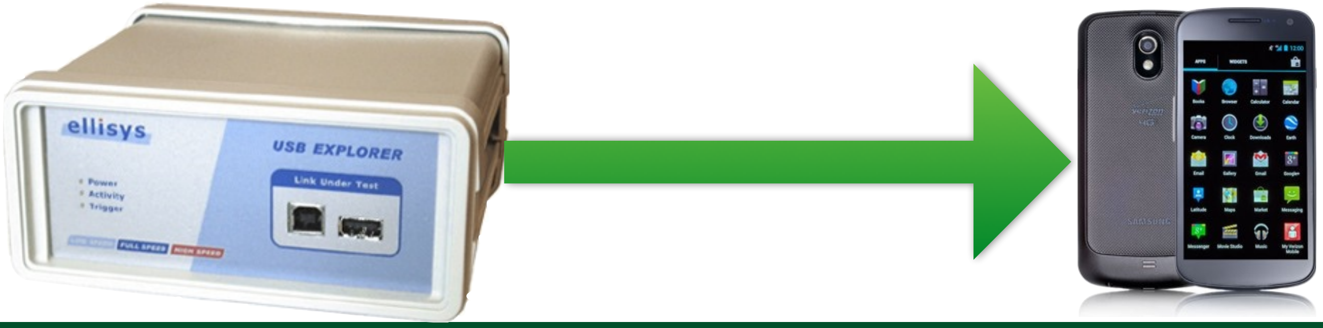


Android USB Analyzer

- Developers use protocol analyzers to debug USB activity as it physically traverses the wire.
- Our Android app for rooted phones loads a modified USB Driver with *kprobe*.

Control Transfer	Android (sec)	Ellisys (sec)
GetDescriptor (Device)	0	0
SetAddress (121)	N/R	0.111 992 200
GetDescriptor (Device)	0.132 202	0.131 983 683
GetDescriptor (Configuration)	0.132 508	0.132 773 683
GetDescriptor (Configuration)	0.132 782	0.132 890 216
GetDescriptor (String: Language)	0.133 057	0.133 015 483
GetDescriptor (String: Product)	0.133 423	0.133 257 716
GetDescriptor (String: Manufacturer)	0.133 698	0.133 390 216
GetDescriptor (String: Serial Number)	0.134 003	0.133 508 250
SetConfiguration	0.134 491	0.133 803 183
GetDescriptor (String: Interface)	0.134 918	0.134 272 716
ClearFeature (Endpoint 1 IN)	N/R	0.165 663 866
ClearFeature (Endpoint 2 OUT)	N/R	0.166 395 233
ClearFeature (Endpoint 1 IN)	N/R	0.167 016 066
GetDescriptor (String: Language)	1.135 742	1.134 755 000
GetDescriptor (String: Serial Number)	1.136 048	1.135 822 016

Buffering delays caused our measurements to lag behind an Ellisys Analyzer by $150\mu s$.



Data Collection

1. Jot down machine attributes in Android app.
2. Hard reset the machine.
3. Disconnect all USB devices from machine.
4. Plug our Android phone into the machine.
5. Android App automatically collected trace data from hundreds of enumerations.

Data Collection

1. Jot down machine attributes in Android app.

2.

3.

4.

Total Data Corpus:

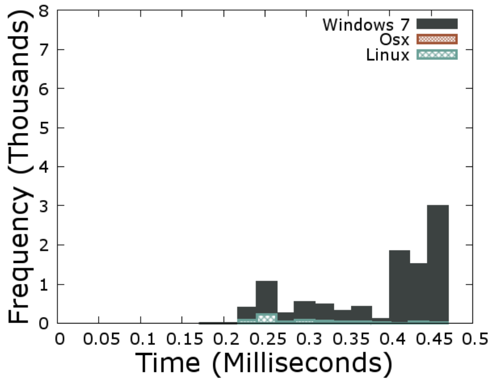
256 Machines Inspected
32,150 Traces Collected

5. Android App automatically collected trace data from hundreds of enumerations.

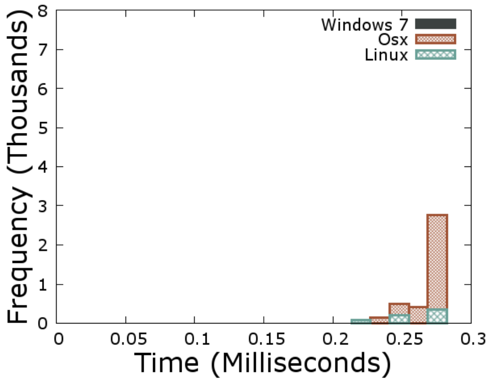
Feature Extraction

Trace Data was processed to extract timing data, producing features that included:

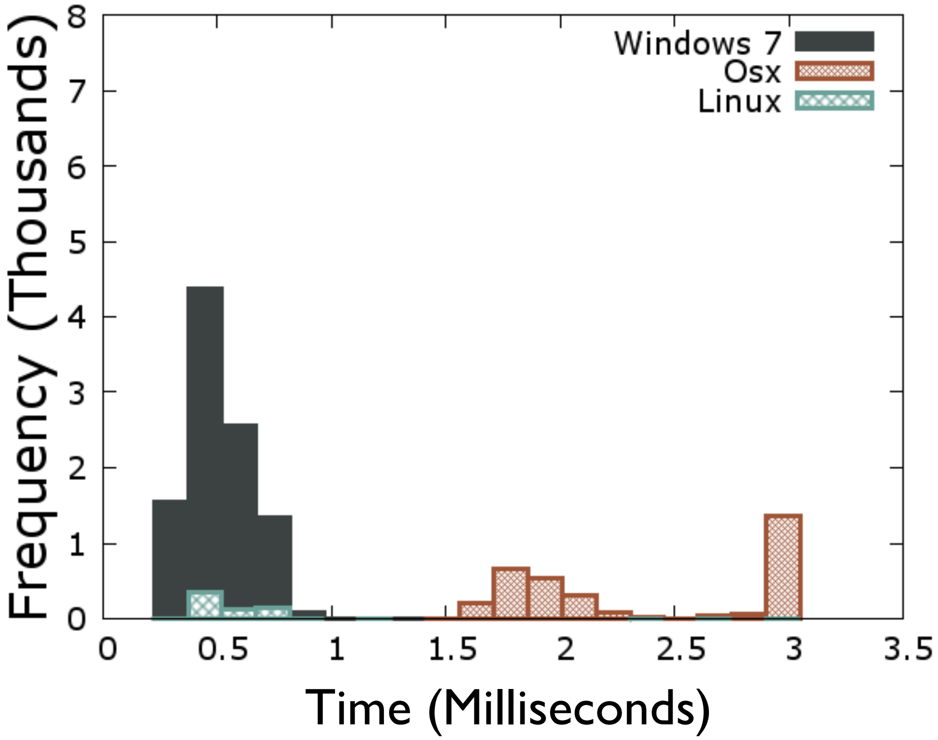
Individual Control Transfers



GET_DESCRIPTOR (Language)

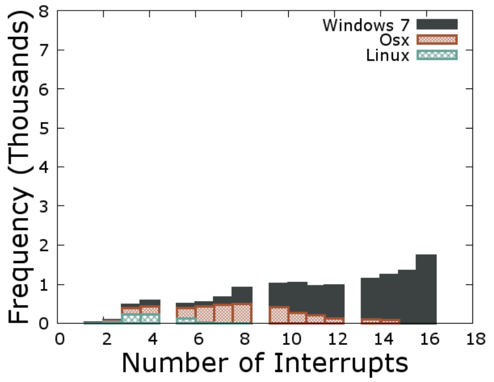


GET_DESCRIPTOR (Manuf.)

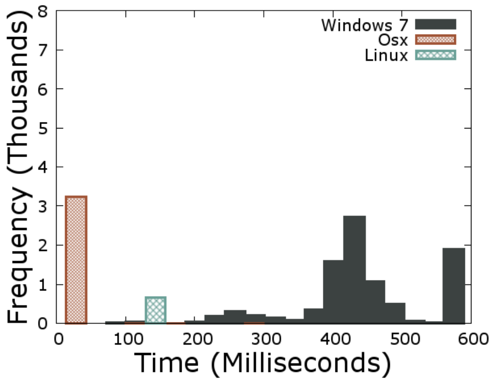


GET_DESCRIPTOR (Serial)

Trace-level Statistics



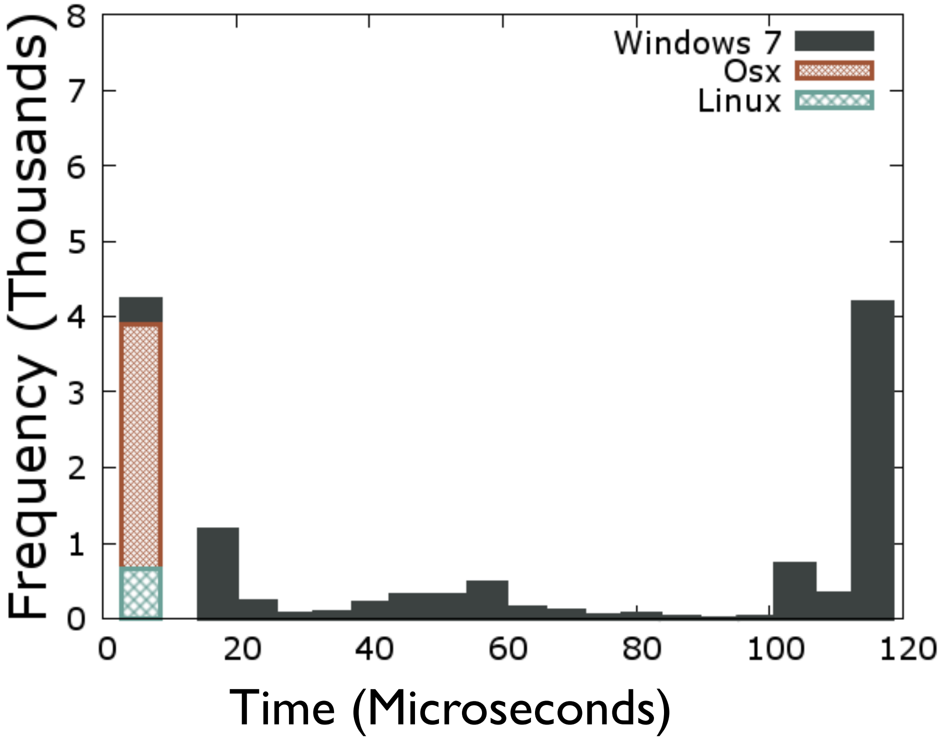
Num. of Idle Packets



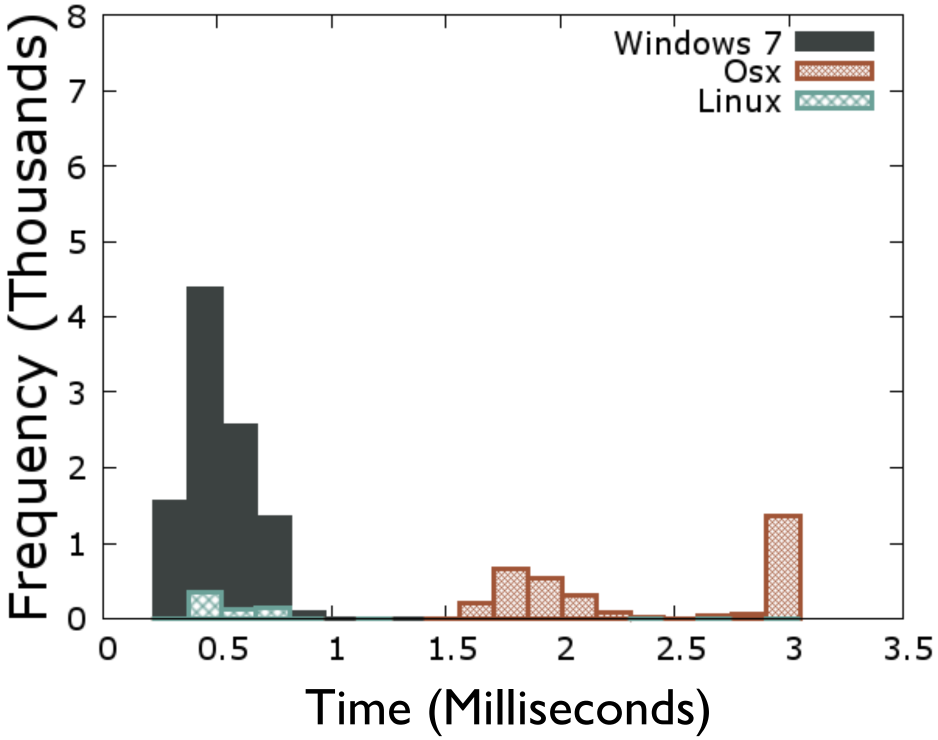
Length of Enumeration

Feature Extraction

Trace Data was processed to extract timing data, producing features that included:



GET_DESCRIPTOR (Int)



GET_DESCRIPTOR (Serial)

Improvements on Past Work

Past schemes have looked at the presence, frequency, and ordering of control transfers to build a scheme. E.g., …

- *"Did the host request the device Manufacturer?"*

  "Must not be a Windows Machine!"

- *"Did the host request the device Language?"*

  "Must not be an OSX Machine!"

- This approach is brittle to URB Spoofing Attacks. We know, because we launched one.

- <u>In contrast, the presence/absence of one feature represents just 0.6% of our feature vector!</u>

Classification

- Supervised learning over vectors of 152 features.
- Considered several labels independently: OS Major Version, OS Minor Version, Model Number.
- Conducted preliminary survey of Weka classifiers, including Random Forests, J48, Boosting, Support Vector Machines (SVM), and Instance Based Learners.
- Selected the *Random Forest* classifier:
  - Models trained on 66% of dataset.
  - Remaining 34% was withheld for evaluation.

Classification Results

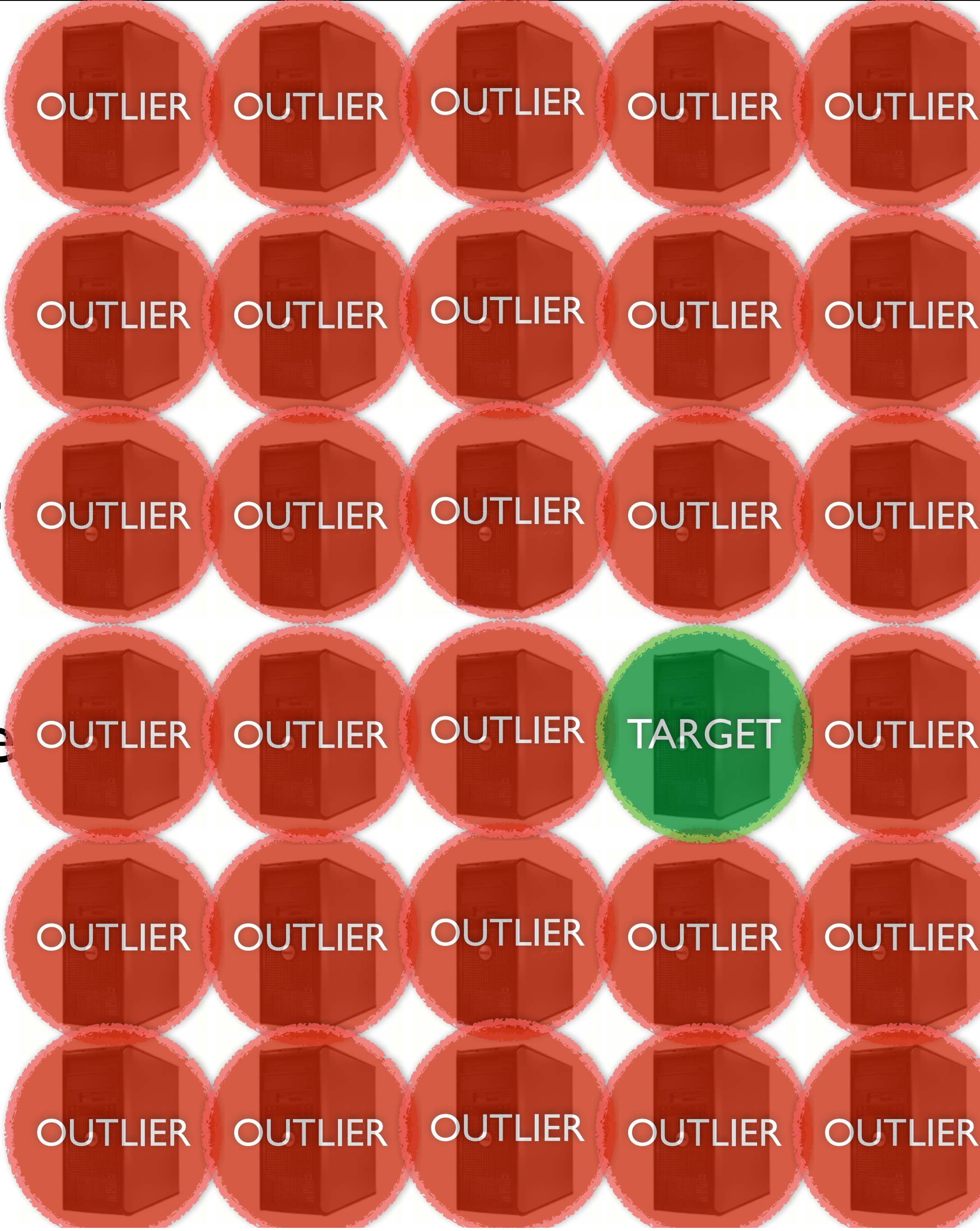
	Example Label	Accuracy
OS Major	“OSX”	100%
OS Minor	“OSX 10.8”	94%
Make	“Apple iMac”	97%
Model	“Apple iMac 13”	90%

Dude, ARE you getting a Dell?

- Field of 30 identical Dell Optiplex 745s.
- Labeled 1 PC "Target", all others "Outlier".
- Polled the classifier for hundreds of predictions.
- Performed χ^2 independence test on the distribution of predictions.
- Tested each PC as Target once.

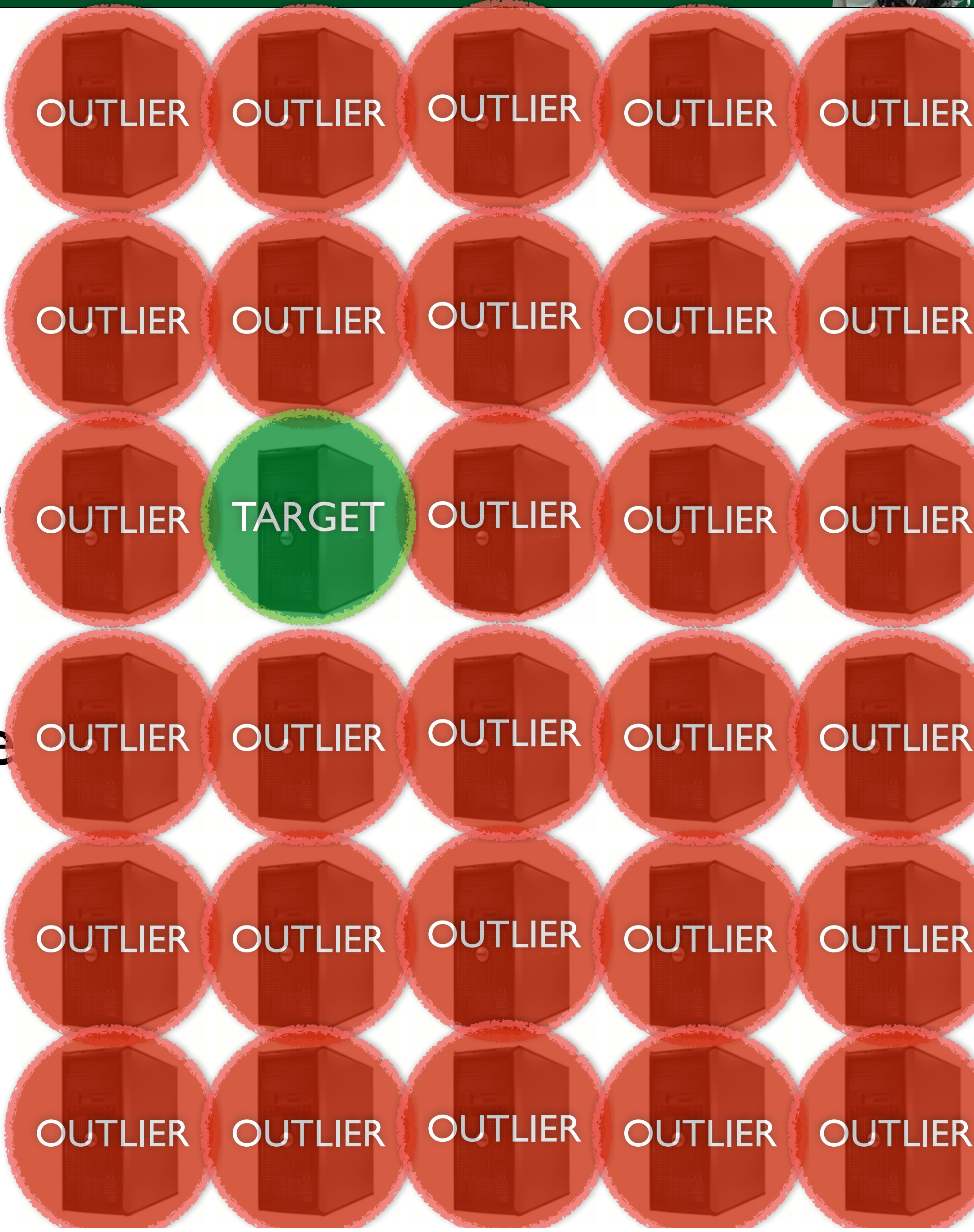
Dude, ARE you getting a Dell?

- Field of 30 identical Dell Optiplex 745s.
- Labeled 1 PC “Target”, all others “Outlier”.
- Classifier predicted label of 100’s of traces.
- Performed χ^2 independence test on the distribution of predictions.
- Tested each PC as Target once.



Dude, ARE you getting a Dell?

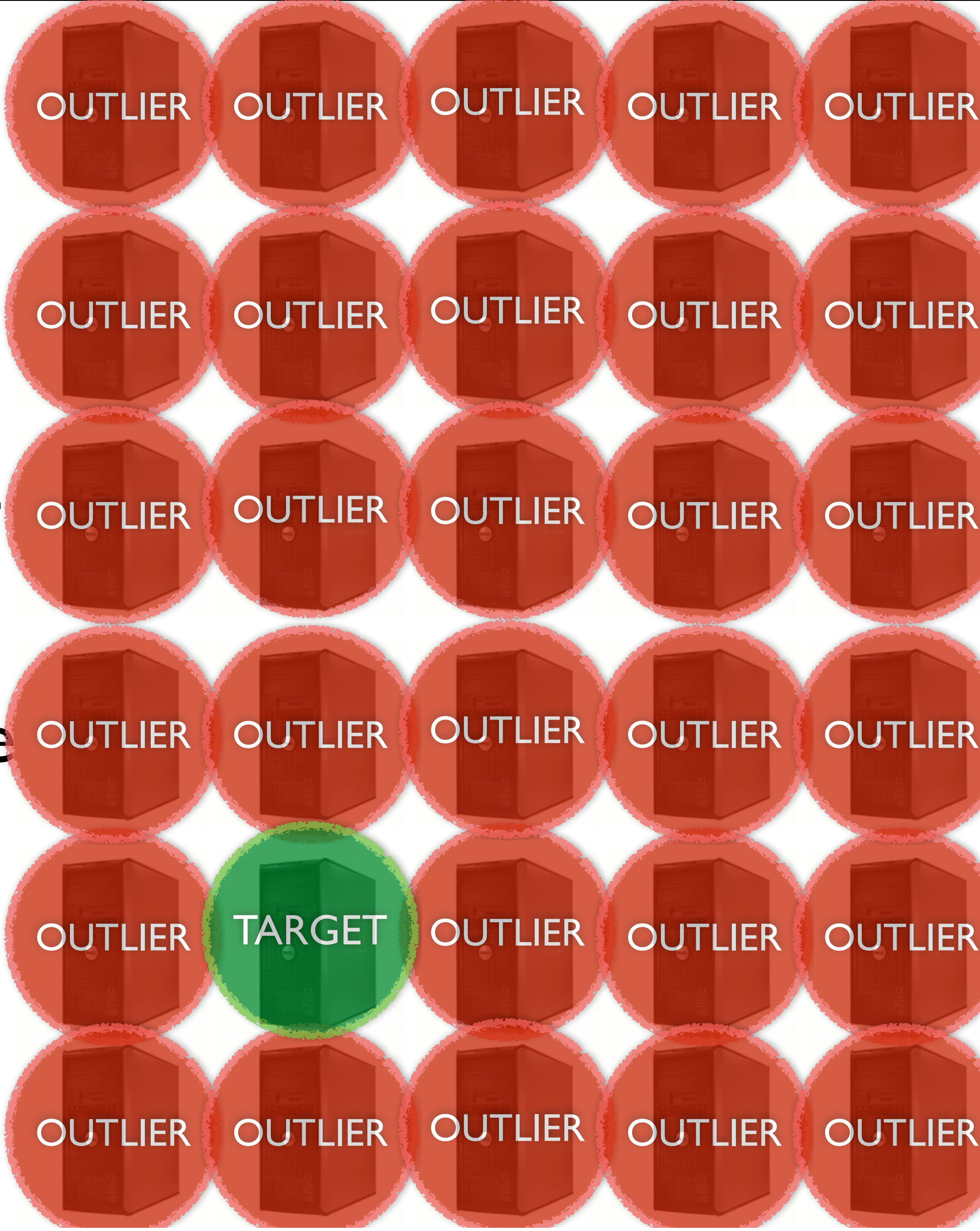
- Field of 30 identical Dell Optiplex 745s.
- Labeled 1 PC “Target”, all others “Outlier”.
- Classifier predicted label of 100’s of traces.
- Performed χ^2 independence test on the distribution of predictions.
- Tested each PC as Target once.



OUTLIER	OUTLIER	OUTLIER	OUTLIER	OUTLIER
OUTLIER	OUTLIER	OUTLIER	OUTLIER	OUTLIER
OUTLIER	TARGET	OUTLIER	OUTLIER	OUTLIER
OUTLIER	OUTLIER	OUTLIER	OUTLIER	OUTLIER
OUTLIER	OUTLIER	OUTLIER	OUTLIER	OUTLIER
OUTLIER	OUTLIER	OUTLIER	OUTLIER	OUTLIER

Dude, ARE you getting a Dell?

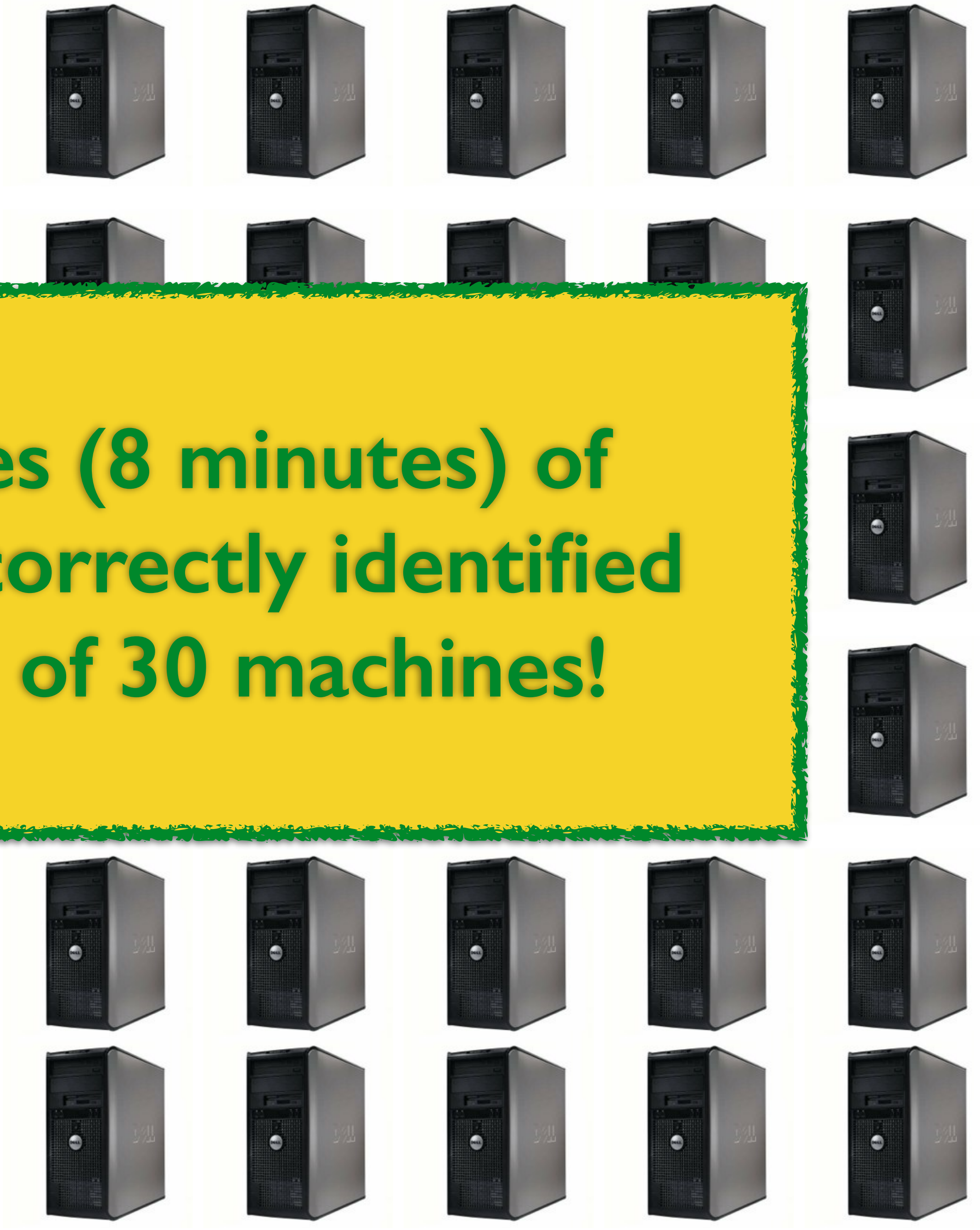
- Field of 30 identical Dell Optiplex 745s.
- Labeled 1 PC “Target”, all others “Outlier”.
- Classifier predicted label of 100’s of traces.
- Performed χ^2 independence test on the distribution of predictions.
- Tested each PC as Target once.



Dude, ARE you getting a Dell?

- Field of 30 identical Dell Optiplex 745s.
- La ot
- Cl 10
- Pe test on the distribution of predictions.
- Tested each PC as Target once.

Given 250 traces (8 minutes) of observation, we correctly identified 70% of the field of 30 machines!



Detect Virtualization

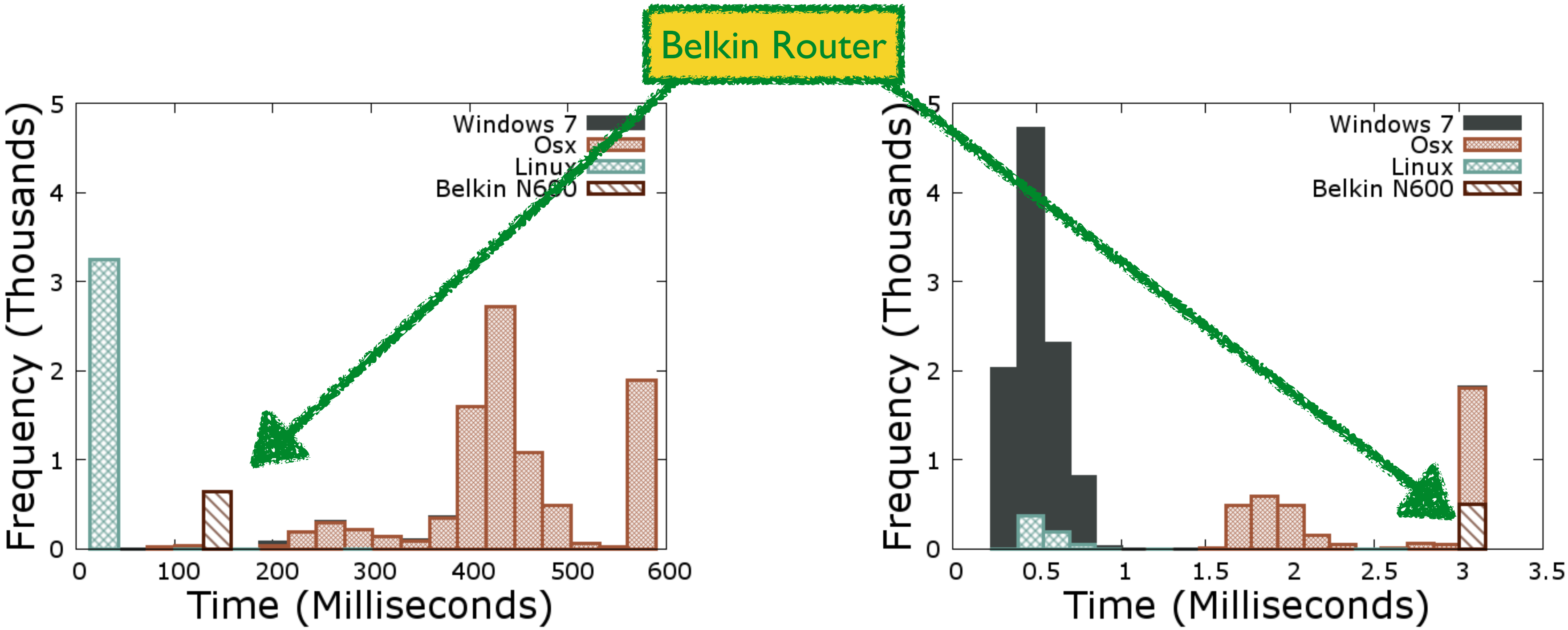
- Can *USB Fingerprinting* detect virtual machine based rootkits??
- Classifier: Anomaly Detection (One Class) SVM with 80% of training data falling into support region.
- Testbed: Train on Bare Metal, Test on Xen dom0.
- Results:

One Trace: 80% acc. (20% F.P., 2.9% F.N.)

50 Traces (13 seconds): 100% acc. in testing!!

Embedded Devices

- We fingerprinted a router, distinguishing it from the rest of our data corpus with 100% accuracy.



Belkin Router

Duration of Enumeration

GET_DESCRIPTOR (Serial)

Security Analysis

Is our fingerprint scheme secure against an adversary that takes an active countermeasure?

- Spoof USB Descriptor Requests… ✓ SECURE
- Send Invalid USB Data… ✗ VULNERABLE
- Launch Relay/Mimicry Attack… ✓ SECURE
- Tamper with USB message timing… ? UNKNOWN

Future Work

UNIVERSITY OF OREGON

- Investigate effects of system load and quiescence
- Improve results for machine identification through
  - Extracting non-timing features
  - Modeling feature distributions over multiple traces
  - USB Fuzzing to prompt unexpected control paths.
- Increase efficacy of *USB Fingerprinting* by supplementing with additional fingerprint techniques.

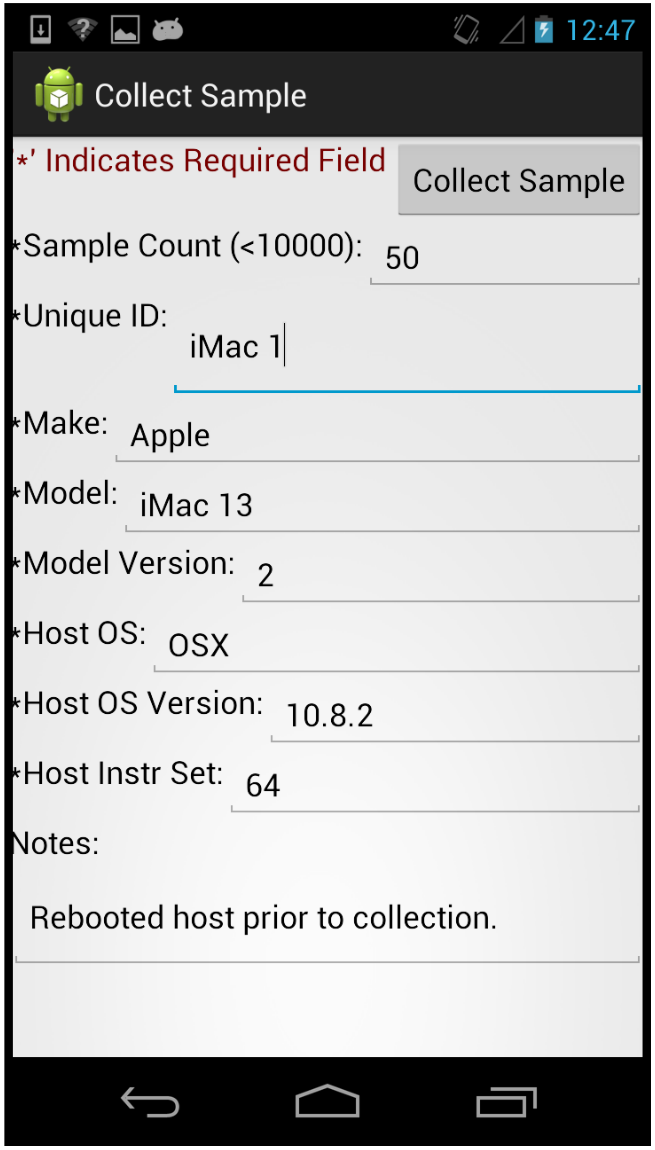
Conclusion

- Shared new results about the information that USB activity leaks about a host machine.
- Demonstrated for the first time that USB timing information can be used to reliably discover the Operating System and Machine Model of hosts.
- Demonstrated for the first time that USB timing information leaks information that can help to distinguish (seemingly) identical machines.

Thank You

Our dataset and source code are now available at

http://osiris.cs.uoregon.edu/



*' Indicates Required Field

Collect Sample

*Sample Count (<10000): 50

*Unique ID: iMac 1

*Make: Apple

*Model: iMac 13

*Model Version: 2

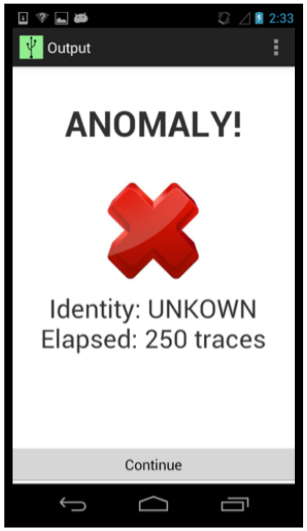
*Host OS: OSX

*Host OS Version: 10.8.2

*Host Instr Set: 64

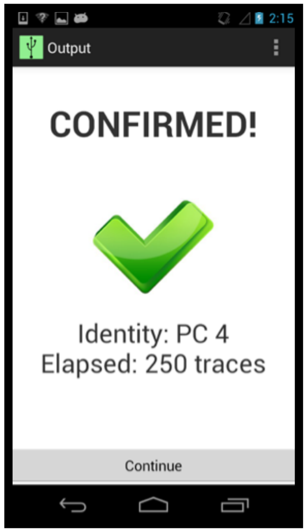
Notes:

Rebooted host prior to collection.



ANOMALY!

Identity: UNKOWN
Elapsed: 250 traces



CONFIRMED!

Identity: PC 4
Elapsed: 250 traces