Towards SDN-Defined Programmable BYOD (Bring Your Own Device) Security

Sungmin Hong, Robert Baykov, Lei Xu, Srinath Nadimpalli, Guofei Gu

SUCCESS Lab
Texas A&M University
Outline

• Introduction & Motivation
• Related Work
• Challenges
• Our Solution PBS (Programmable BYOD Security)
• Evaluation
• Conclusion
Bring Your Own Device

• BYOD is the *new paradigm* in the workplace
  • 44% of users in developed countries and 75% in developing countries are now utilizing BYOD in the workplace¹
  • The adoption rate shows no signs of slowing
    • Surveys have indicated that businesses are unable to stop employees from bringing personal devices into the workplace²

![Image source: www.itproportal.com](http://www.itproportal.com)

Admins’ Headache

I need to set what apps are allowed at work. But during the whole work hours?

Allow an email app any time and Facebook at lunch time.

Anywhere in the workplace?

I want to let a visitor access to the Internet through different VLAN. Anywhere in the workplace?

I need to monitor what apps access to our database.

Apply it for any one? I want to restrict the rule by role. What if an employee turn off WiFi and turn on LTE to enjoy Twitter at the bathroom?

What if the policy is changed?
Admins’ Concerns

• Ideally,
  • Manage & control BYOD devices easily, efficiently, and securely
  • Less budget expense

• However,
  • Management of dynamic BYOD-enabled devices become significantly more complex
  • Diverse (biz/non-biz) apps to monitor
  • Network itself needs more security and management capabilities to protect enterprise resource
  • Additional infrastructure required
Motivation of Our Work

- **Application Awareness & Network Visibility**
  - App-aware network information & user/device contexts are invisible to traditional tools/infra.
  - App may send data through other network interfaces (e.g., 3G/4G) equipped in the device
  - Correlating app’s network activities with the contexts is not easy

- **Dynamic Policy Programming**
  - Static access/policy control is not sufficient for network/BYOD dynamics for finer-grained management
Outline

• Introduction & Motivation
• Related Work
• Challenges
• Our Solution PBS (Programmable BYOD Security)
• Evaluation
• Conclusion
Related Work

• Google
  • Android Device Administration (ADA)
    • Device-level control on password, remote device wiping, etc.
    • Limited interfaces and features
  • Android for Work (AFW)
    • “WorkProfile” to separate enterprise and personal app data
    • OS-level encryption and additional management APIs to third-party MDM/Enterprise Mobility Management (EMM) partners
    • Focus on device/app data control and protection
    • Limited functionalities to support dynamic context-aware policy enforcement

• Samsung KNOX
  • Enterprise container to separate enterprise and personal app data
  • H/W-level encryption and management APIs to EMM partners
  • Dedicated device only
  • Limited functionalities to support dynamic context-aware policy enforcement
Related Work

• Mobile Device Management (MDM)
  • Provide additional granularity and complexity in management capabilities through ADA (normally through proprietary hardware)
  • Requires additional infrastructure and network reconfiguration

• Android research
  • DeepDroid
    • Enforce app & context-aware policies to protect sensitive on-device resource by tracking the system APIs
    • Less fine-grained policy configuration
    • Lack programmable interfaces for dynamic, reactive policy enforcement

➔ We provide a solution in our work to these shortcomings
Outline

• Introduction & Motivation
• Related Work

• Challenges
• Our Solution PBS (Programmable BYOD Security)
• Evaluation
• Conclusion
Research Challenges

• Can we use traditional security solutions?
  • Difficult and inflexible for dynamic, N/W- and app-aware security policy enforcement (e.g., ACLs/firewalls)
  • Typically coupled with physical devices/resources instead of applications

• Can we apply the legacy SDN infrastructure?
  • Additional cost to build/manage the infrastructure (e.g., OpenFlow-enabled switches)
  • Lack of BYOD specifics
    • App & context unaware
    • Loss of global visibility from other on-device network interfaces (3G/4G, BT, etc.)

• How much granularity we should provide?
  • The finer granularity (from layer 2, app & context-aware), the more useful to security policy enforcement
Outline

• Introduction & Motivation
• Related Work
• Challenges
• Our Solution PBS (Programmable BYOD Security)
• Evaluation
• Conclusion
PBS (SDN-Defined Programmable BYOD Security)

• Goals and Contributions

• Fine-grained Access Control
  • Application & context-aware access control with layer2 and above granularity

• Dynamic Policy Enforcement
  • Dynamic, reactive policy enforcement at run-time based on application-specific policy and network behavior

• Network-wide Programmability
  • Programmable network-wide policy enforcement system to enterprise admin

• Minor Performance Overhead
  • Minimize performance overhead and resource consumption for mobile devices

• No Additional Infrastructure
  • On-device SDN-based solution without deploying additional OpenFlow switches
Basic Idea (1/2)

- Abstraction inside the device
  - App & Context awareness + Visibility
  - SDN-transparent flow management
  - No infrastructure required
Basic Idea (2/2)

• Dynamic Programmability
  • SDN-based Network Programming Capabilities with:
    • App & Context awareness + Visibility
    • Policy language

PBS Client

PBS Controller (SDN-based)
Operations

• Application-aware flow control
Operations

- Visibility (No hidden network)
Operations

- Proactive Policy Enforcement
Operations

• Dynamic & Reactive Policy Enforcement
Operations

• Real-time Context
Operations

- Tailored to Mobile Environment
  - Minimize the controller intervention
  - Optimize app & context aware flow management

Mobile App

**PBS Client**

- Message PushDown
- Two-tiered Programming
- Short-circuit

Net Inf.

- WiFi
- 3G/4G
- BT

**PBS App**

- PBS Controller

**Internet**

- Business Server
- Security Middlebox
- Enterprise Network
- WiFi
- BT
- 3G/4G
Operations

• High-level Policy Language
  • Makes policy definition simple without requiring expert knowledge on SDN.

```
Target := APP (APP_ID | APP_NAME | ALL) |
          APP_GRP (TRUST | THIRD_PARTY | UNKNOWN) |
          DEVICE (DEV_ID | GROUP | UNAUTHORIZED | ALL)
Match := OF_MATCH
Predicate := {Event + Condition}
Event := PORT_STAT | LOC | TIME |
        USR_ROLE | DEV_MODE | CNTRL_STATE |
        PKT | RATE
Condition := {Operator + Value}
Value := DECIMAL_NUMBER
Actions := Control | Manage | Trigger
Control := ALLOW | DENY |
          {REDIRECT | MIRROR | QUARANTINE} +
          ADDR (IP | CONTROLLER)
Manage := REPORT | OF_ACTION
Trigger := IMMEDIATE | PERIODIC + Value
```
Operations

• Policy Example 1

<Policy PolicyID=Employee>
  <Target app=com.facebook.android app_grp=THIRD_PARTY>
    <Match>nl_dst=66.220.144.0</Match>
    <Predicate>USR_ROLE=Business,TIME ge 0800,TIME le 1800</Predicate>
  </Target>
  <Actions>
    <Control>REDIRECT=CONTROLLER</Control>
    <Manage>REPORT</Manage>
    <Trigger>IMMEDIATE</Trigger>
  </Actions>
</Policy>

• Policy Example 2

<Policy PolicyID=All_Unauth_Dev>
  <Target device=UNAUTH app=ALL>
    <Match>*</Match>
    <Predicate>TIME ge 0800,TIME le 1800</Predicate>
  </Target>
  <Actions>
    <Control>REDIRECT=123.45.67.8</Control>
    <Manage>OF_ACTION(set_vlan_id)=UNAUTH_VID</Manage>
    <Trigger>IMMEDIATE</Trigger>
  </Actions>
</Policy>
Outline

• Introduction & Motivation
• Related Work
• Challenges
• Our Solution PBS (Programmable BYOD Security)
• Evaluation
• Conclusion
Evaluation

• Performance Overhead
  • Testing Environment
    • LG Nexus 5 with a Qualcomm MSM8974 Snapdragon 800 CPU
    • Asus Nexus 7 tablet with an ARM Cortex-A9
    • Both run Android system version 4.4 (KitKat)
    • Controller runs on Ubuntu Linux x64 with a Quad Core CPU with 8 GB RAM
  • Benchmark tools used for the evaluation:
    • Iperf, Antutu, Geekbench, Vellamo, and PCMark
Performance

• Network Throughput Benchmark
  • Test duration as 10 minutes with a two-second interval between periodic bandwidth reports.

• Battery Overhead (PCMark) *(Note that lower is better)*

<table>
<thead>
<tr>
<th>Attribute</th>
<th>NX5</th>
<th>NX5 PBS</th>
<th>Over.</th>
<th>NX7</th>
<th>NX7 PBS</th>
<th>Over.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browsing</td>
<td>3741</td>
<td>3926</td>
<td>4.95%</td>
<td>1972</td>
<td>2176</td>
<td>10.34%</td>
</tr>
<tr>
<td>Writing</td>
<td>3174</td>
<td>3436</td>
<td>8.25%</td>
<td>2202</td>
<td>2301</td>
<td>4.50%</td>
</tr>
<tr>
<td>Video</td>
<td>4118</td>
<td>4276</td>
<td>3.84%</td>
<td>3739</td>
<td>3893</td>
<td>4.12%</td>
</tr>
<tr>
<td>PhotoEdit</td>
<td>4804</td>
<td>4948</td>
<td>2.99%</td>
<td>2591</td>
<td>2597</td>
<td>0.23%</td>
</tr>
<tr>
<td>Total</td>
<td>15837</td>
<td>16586</td>
<td>4.73%</td>
<td>10504</td>
<td>10967</td>
<td>4.41%</td>
</tr>
</tbody>
</table>
## Performance

• System Performance Benchmark

### Nexus 5

<table>
<thead>
<tr>
<th>Type</th>
<th>Benchmark</th>
<th>NX5 PBS</th>
<th>NX5</th>
<th>Overhead %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>Antutu</td>
<td>31824</td>
<td>33600</td>
<td>5.3</td>
</tr>
<tr>
<td></td>
<td>Vellamo</td>
<td>3009</td>
<td>3044</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>PCMark</td>
<td>15201</td>
<td>16122</td>
<td>5.7</td>
</tr>
<tr>
<td></td>
<td>Geekbench</td>
<td>2994</td>
<td>3185</td>
<td>6.0</td>
</tr>
<tr>
<td>CPU</td>
<td>Vellamo</td>
<td>1599</td>
<td>1644</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td>Geekbench</td>
<td>6349</td>
<td>6744</td>
<td>5.9</td>
</tr>
<tr>
<td>RAM</td>
<td>Antutu</td>
<td>2199</td>
<td>2295</td>
<td>4.2</td>
</tr>
<tr>
<td></td>
<td>Geekbench</td>
<td>2323</td>
<td>2440</td>
<td>4.8</td>
</tr>
</tbody>
</table>

### Nexus 7

<table>
<thead>
<tr>
<th>Type</th>
<th>Benchmark</th>
<th>NX7 PBS</th>
<th>NX7</th>
<th>Overhead %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>Antutu</td>
<td>17822</td>
<td>18076</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>Vellamo</td>
<td>1524</td>
<td>1609</td>
<td>5.3</td>
</tr>
<tr>
<td></td>
<td>PCMark</td>
<td>10937</td>
<td>11187</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td>Geekbench</td>
<td>1363</td>
<td>1435</td>
<td>5.0</td>
</tr>
<tr>
<td>CPU</td>
<td>Vellamo</td>
<td>1016</td>
<td>1095</td>
<td>7.3</td>
</tr>
<tr>
<td></td>
<td>Geekbench</td>
<td>3233</td>
<td>3413</td>
<td>5.3</td>
</tr>
<tr>
<td>RAM</td>
<td>Antutu</td>
<td>2252</td>
<td>2269</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>Geekbench</td>
<td>353</td>
<td>354</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Use Cases

• Use Case 1: Network Activity Logging
  • netlog
    • Global visibility of app-aware flows
    • Network behavior monitoring
    • Configuration validation
    • Security audit

• Use Case 2: Network Policy Enforcement
  • netPol
    • Dynamic, reactive network policy
    • Real-time context-specific programmability

• Use Case 3: App Flow Path Management
  • netBal
    • Traffic redirection for network load management
    • Security management
    • Isolation, redirection, quarantine,
Outline

• Introduction & Motivation
• Related Work
• Challenges
• Our Solution PBS (Programmable BYOD Security)
• Evaluation
• Conclusion
Conclusion

• We propose a new network security framework for BYOD, PBS (Programmable BYOD Security)

• We achieve dynamic, fine-grained network control of applications on mobile devices

• With PBS, administrators also benefit from the global network visibility and fine-grained policy programmability

• Without imposing much performance overhead, PBS-DROID can effectively enforce the dynamic network access control policy with users’ context information.
Thank You