Symantec.

# All Your Root Checks Are Belong to Us

Azzedine Benamevr, Nathan Evans, Yun Shen

# Agenda

- Bio
- Context
- What is root ?
- Reversing toolbox
- Examination of common root detection methods
- Security and BYOD apps
- AndroPoser
- Conclusion

# All your Root Checks are Belong to Us: Bio

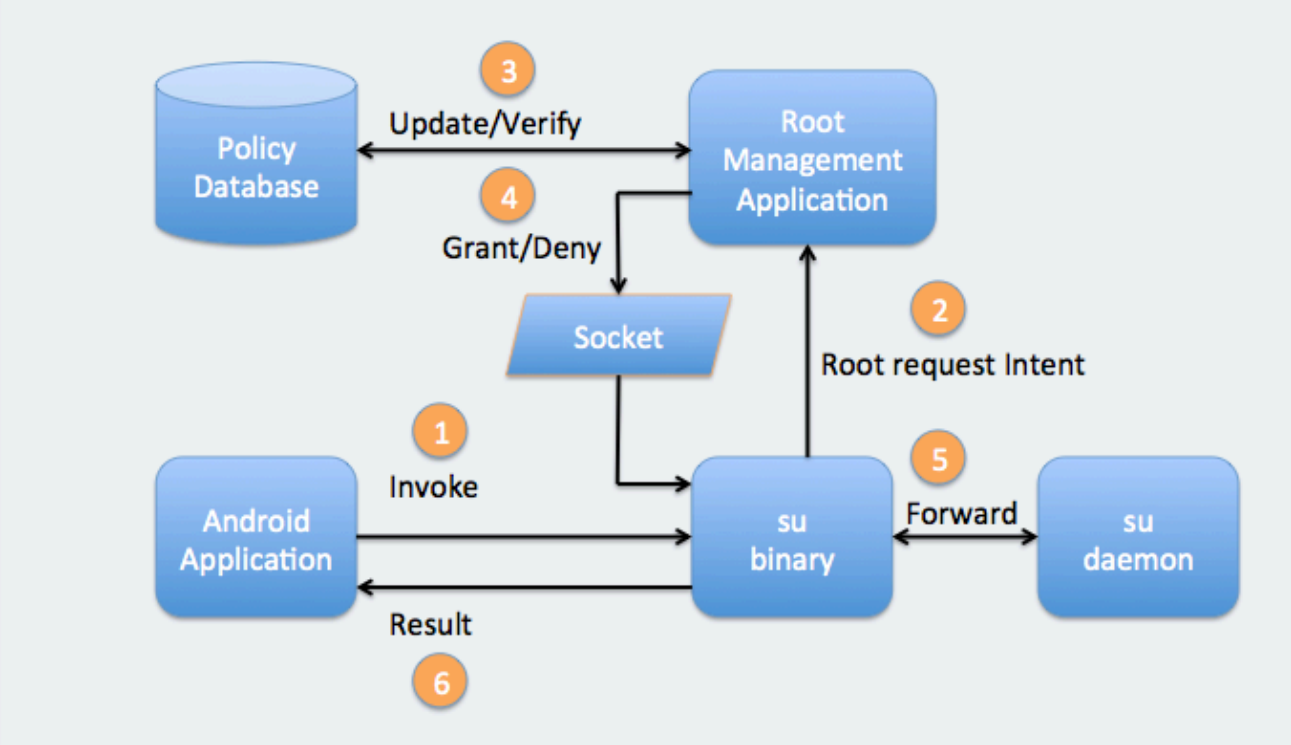
- Azzedine Benameur:
  - Joined Symantec in 2011
  - Past research projects: Minestrone (IARPA) and MEERKATS (DARPA)
  - SAP Security&Trust Lab, HP Cloud Security Lab
  - Ph.D., Computer Science from Lyon University, 2009
- Nathan Evans:
  - Joined Symantec in 2011
  - Past research projects: Minestrone (IARPA) and MEERKATS (DARPA)
  - AFRL-funded research in network security/mapping (NICE)
  - Ph.D., Computer Science from T.U. Munich, 2011
- Yun Shen:
  - Joined Symantec in 2012
  - Past research project: Bigfoot (FP7)
  - HP Cloud Security Lab
  - Ph.D., Computer Science from University of Hull, 2005

# All your Root Checks are Belong to Us: Context

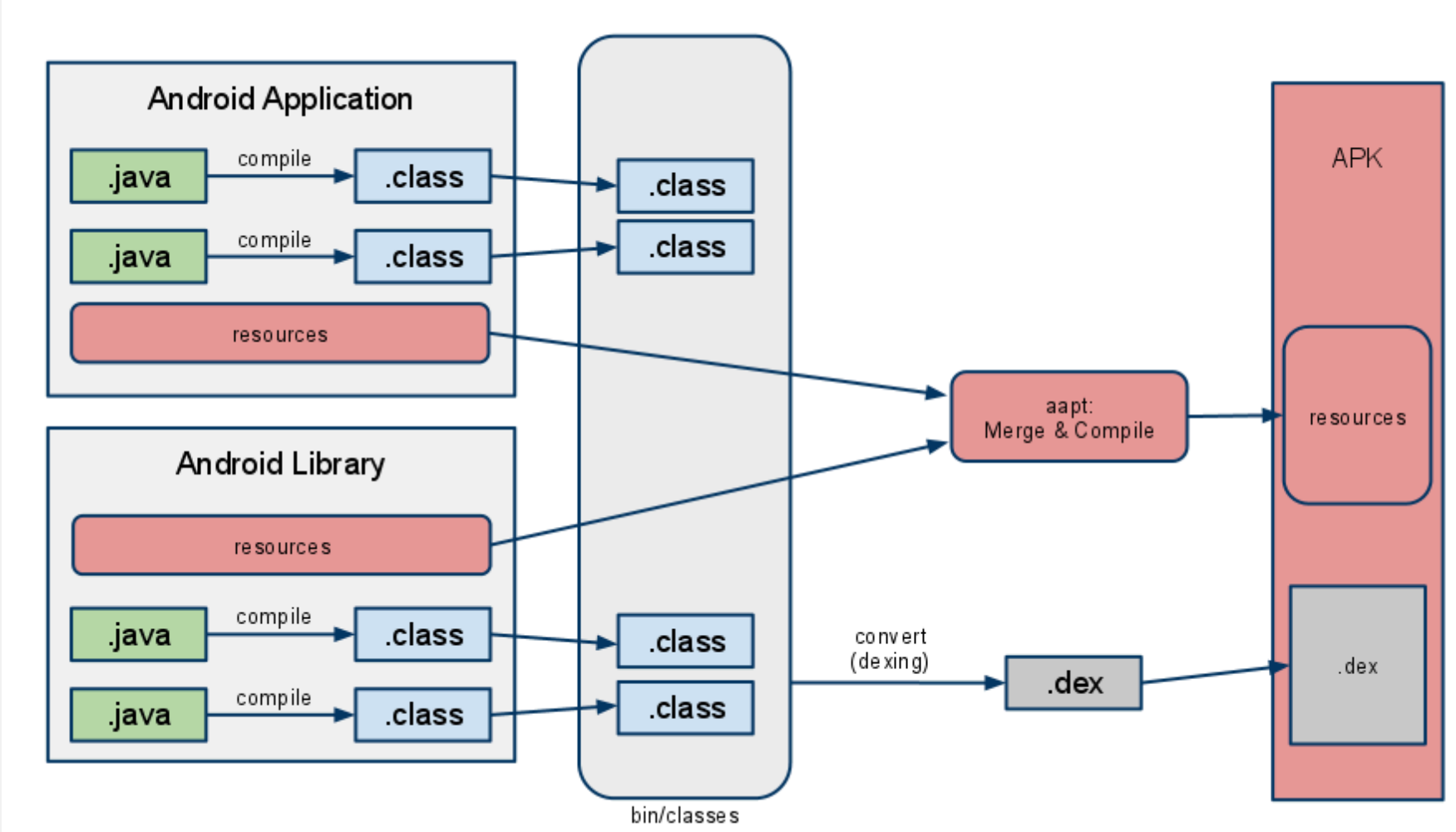
- The rise of BYOD:
  - Personal and corporate data on the same device
  - Android is on 70% of the devices (phone and tablet)
  - Root has a bad reputation
- Questions
  - How many/which applications check for root as a security concern?
  - How are these checks implemented/are they effective at detecting root?
  - How easily can these checks be subverted to hide the presence of root?
  - What are the implications?
- Methodology
  - Look at the top security/MDM solutions available
  - Compare root detection methods based on standard set of checks

# All your Root Checks are Belong to Us: What is root ?

How Root works:

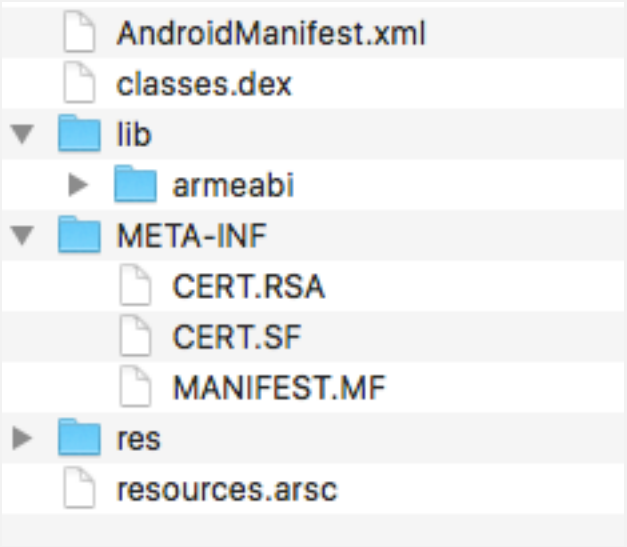


# All your Root Checks are Belong to Us: ToolBox



# All your Root Checks are Belong to Us: ToolBox

- Android Application are distributed as APK:



# All your Root Checks are Belong to Us: ToolBox

- **Dex2Jar**: converts android bytecode to Java Archive (JAR)
- **JD-core:** converts JAR to Java source code.
- **Apktool**: decompiles android byte code to an intermediate language (in case the Java source code was not fully recovered or the analysis was inconclusive)
- Custom Scripts: automate the process and search for obvious Java calls and broad references to rooted phone features

# **All your Root Checks are Belong to Us:** Common Root Discovery

- Presence of files:
  - **Static PATH:** Hardcode paths (/system/bin/su, /bin/su, etc.) and issue an open/stat
  - **Dynamic PATH:** Parse the PATH variable, appending "/su" to each entry; open each in a loop
  - **System PATH**: Executes which command with parameter "su" and check if the result is 0
  - **Execution:** Just attempt to execute "su" as a subprocess and check the return code
  - **Root ACL Program**: Check for superuser apk under the path "/system/app/Superuser.apk".
  - **Setuid:** We found one app with an interesting check; the presence of binaries on the system that were setuid root, or able to be executed as root (uid 0) by normal users. While standard su binaries are setuid root, we are not sure if this is a legitimate check for root as programs could be setuid root for other reasons.
  - **Installed Packages:** Check for the presence of common root packages being installed on the system (e.g., "com.chainfire.supersu","com.noshufou.android.su"). We saw both checks using Android API's as well as by exec'ing "pm list packages"

# **All your Root Checks are Belong to Us:** Common Root Discovery

- General Device Settings:
  - **Test keys:** If a custom kernel is used on a device the build version shows that “test-keys” are used instead of “release-keys”. Some apps assume “test-keys” means the device is rooted, which is not always the case. Also, the presence of “release-keys” does not indicate the device is not rooted.
  - **Build version:** We encountered specific checks of the setting “ro.modversion” as well, which can be used to identify certain custom Android ROMs (such as Cyanogenmod).

# **All your Root Checks are Belong to Us:** Common Root Discovery

- Runtime Capabilities and Characteristics:
  - **System mounted:** Some rooting methods require this partition to be remounted “rw” (read- /write). We saw two variants of this check; the first simply runs the mount command and looks for a “rw” flag, the second actually attempts to create a file under “/system/” or “/data/”.
  - **Ability to mount:** A related method attempts to mount the “/system” partition with the command “*mount -o remount,rw /system*”, and then checks the return code.
  - **User ID:** A curious check we found in one case was the app getting the current user id (UID) of the app as it was running and checking if it was running as root (UID 0). This is curious because as far as we know, even on a rooted phone any app started by Zygote gets it’s own unique (non 0) UID. However, it is possible that an app would request root access via intent and then issue the UID check.

# All your Root Checks are Belong to Us:
## Security Applications

| Company Name | Static su | Relative su | Test Keys | ACL Prog | App List | Mount /system | UID 0 | Total Checks | App Installs |
|---|---|---|---|---|---|---|---|---|---|
| AVAST | | ✓ | | | | ✓ | | 2 | 100M-500M |
| Lookout | ✓ | | ✓ | ✓ | | | | 3 | 100M-500M |
| Cleanmaster | ✓ | | | | | | | 1 | 100M-500M |
| Qihoo | | ✓ | | | | | | 1 | 100M-500M |
| AVG | ✓ | | | | | | | 1 | 100M-500M |
| Mcafee | | | | | | | | 0 | 10M-50M |
| Norton | ✓ | | | | | | | 1 | 10M-50M |
| NQ Labs | ✓ | ✓ | | | | ✓ | | 3 | 10M-50M |
| Kaspersky | | ✓ | ✓ | ✓ | | ✓ | | 4 | 10M-50M |
| Trustlook | | ✓ | ✓ | ✓ | | | | 3 | 10M-50M |
| Avira | ✓ | | ✓ | ✓ | | | | 3 | 10M-50M |
| Trend | | ✓ | | | | | ✓ | 2 | 1M-5M |
| ESET | | ✓ | ✓ | ✓ | | | | 3 | 1M-5M |
| CY Security | ✓ | | | | | ✓ | | 2 | .5M-1M |
| Panda | | | | | | | | 0 | .5M-1M |
| Sophos | | | | | | | | 0 | .1M-.5M |

# **All your Root Checks are Belong to Us:** Security Applications

- No native code used for root detection ?!
- RootCloak/Xposed “friendly”
- AVAST leverages root: iptables/firewall
- Kaspersky root checks: packed/reflection (required runtime help)

# All your Root Checks are Belong to Us: BYOD Solutions

| Company Name | Static su | Relative su | Test Keys | ACL Prog | App List | Mount /system | UID 0 | Total Checks | App Installs |
|---|---|---|---|---|---|---|---|---|---|
| MobileIron | | | | | | | | 0 | 1M-5M |
| VMware | | ✓ | | ✓ | ✓ | | ✓ | 4 | 1M-5M |
| Kaspersky | | ✓ | ✓ | ✓ | | ✓ | | 4 | 500k-1M |
| Citrix | ✓ | | ✓ | ✓ | | | | 3 | 100k-500k |
| IBM | ✓ | ✓ | ✓ | ✓ | ✓ | | | 5 | 100k-500k |
| SAP | ✓ | ✓ | | ✓ | | | | 3 | 100k-500k |
| McAffee | ✓ | ✓ | | ✓ | | | | 3 | 100k-500k |
| Excitor* | ✓ | | ✓ | ✓ | | | | 3 | 50k-100k |
| AVG | ✓ | | | | | | | 1 | 10k-50k |
| Symantec | ✓ | | | | | | | 1 | 10k-50k |
| Deutsche TK | | | | | | | | 0 | 10k-50k |
| GLOBO | | ✓ | | ✓ | | | | 1 | 10k-50k |
| Tangoe | ✓ | | | ✓ | | ✓ | | 3 | 10k-50k |
| Soti | ✓ | | ✓ | | | | | 2 | 10k-50k |
| Amtel | ✓ | | ✓ | ✓ | | | | 3 | 5k-10k |
| Dell | ✓ | | ✓ | ✓ | | | | 3 | 5k-10k |
| Wavelink | | | | | | | | 0 | 5k-10k |
| Good | ✓ | | ✓ | ✓ | ✓ | | | 4 | 1k-5k[1] |
| Panda | | | | | | | | 0 | 1k-5k |

# **All your Root Checks are Belong to Us:** BYOD Solutions

- **Native code:**
  - VMware's Airwatch MDM agent. *libcoredevice.so* is not particularly difficult to reverse. The bulk of the checks are in the method *getDeviceState(JNIEnv *, jobject *)*
  - Excitor: The library is not difficult to reverse: static path along with privilege escalation attempt.
- **Four vendors have no root check:**
  - MobileIron: *com.cisco.anyconnect.vpn.android.rooted*, Policy might be pushed from server later ?
- **Breadth:** We were impressed by the apparent effort that went into making IBM's MDM solution as rigorous and in-depth as possible. But no obfuscation and no native code

# **All your Root Checks are Belong to Us:** AndroPoser

- Needed a tool to verify our static analysis
- Created a library that gives us runtime visibility into what the process is really doing
- Leveraged dynamic linker feature: LD_PRELOAD
- Selected a set of functions to inspect

# **All your Root Checks are Belong to Us:** AndroPoser

- Easy to set on Android:
  - setprop wrap.com.package.id “LD_PRELOAD=/data/androposer.so”
- Example on open():

```
int  open(const char*  path, int  mode, ...)
{
  int ret;
  int (*real_open) (const char *, int, ...);
  real_open = dlsym(RTLD_NEXT, "open");
  __android_log_print(ANDROID_LOG_DEBUG,
        "AndroPoser: OPEN", "Path: {%s}", path);
  if (strcmp(path, "/system/bin/su") ==0)
        return -1;
  ret = real_open(path, mode);
  return ret;;
}
```

# **All your Root Checks are Belong to Us:** AndroPoser

- Other use: defeat Anti-Debug protection
- Identify FD for open /proc/self/status

```
real_open = dlsym (RTLD_NEXT, "open");
if (0 == strcmp (path, "/proc/self/status"))
{
  ret = real_open (path, flags);
  status_fd = ret;
  write_to_log ("Open for status called, setting status fd `%d'\n", status_fd);
}
```

- Replace read for “TracerPid: XXX” (where XXX is the debugger)

```
if ((status_fd != 0) && (fd == status_fd))
{
  if ((tracer_off = strstr ((char *)buf, "TracerPid:\t")) != NULL)
  {
    tab_off = strstr (tracer_off, "\t");
    if (tab_off != NULL) /* Should never happen */
    {
      tab_off[1] = '0';
      tab_off[2] = '\n';
    }
  }
}
```

# **All your Root Checks are Belong to Us:** Conclusion

- Security/MDM comparison:
  - McAfee: No check on their AV but checks on the MDM agent
  - Kaspersky/Symantec: Same code for both security and MDM app, Kaspersky has a different build probably
  - Panda: no apparent root check for either
- BYOD/MDM solutions do care about Root
- Most are vulnerable to RootCloak/Xposed/AndroPoser

| Root Check Type | Andro-Poser | Root-Cloak | Hide My Root |
|---|---|---|---|
| Static Path | ✓ | ✓ | ✓ |
| Relative Path | ✓ | ✓ | ✓ |
| Native Code | ✓ | NO | NO |
| ACL Prog | ✓ | ✓ | NO |
| UID | ✓ | NO | NO |

# **All your Root Checks are Belong to Us:** Conclusion

- Level up:
  - We revisited how we check for root
  - Native code is making it a little bit harder
  - Binary “hardening”: packing, check-summing, string encryption

- Exploring other ways:
  - Machine learning based approach to detect root: WIP
  - ARM TrustZone ?

# Thank you!