Extracting and Examining SoC Firmware from Amazon Echo's embedded Multi-Media Card (eMMC)

Modern embedded devices such as the Amazon Echo are fundamentally similar to a typical computer. They contain a CPU, some sort of input and output (I/O), and a memory interface(s). eMMC is a memory format similar to a solid state drive (SSD) in which data can be loaded to it and persist across power cycles. eMMC is a compact and relatively inexpensive chip, which is why it is commonly used in mobile phones and embedded devices to store the device's firmware. This practice is commonly referred to as System-on-Chip (SoC), in which some or all of a computer's components are integrated into one integrated circuit.

This demonstration will show how eMMC packages can be removed from an embedded device, loaded into an adapter, and its contents extracted. In doing so, the embedded firmware will be examined, ideally to reveal sensitive system information.

Disassembly

When taking apart the "Amazon Echo Dot 2nd Generation", various components are revealed, primarily the "SKhynix H9TQ32A4GTMC BGA221" eMMC Flash Memory Chip, highlighted in red in Figure 1.3.



Figure 1.1 - Amazon Echo Dot Gen II

Figure 1.2 - Amazon Echo Dot Gen II Teardown



Figure 1.3 - Amazon Echo Dot Gen II eMMC Flash Memory Package



In order to be able to load the eMMC chip into an adapter and read its contents, it must first be removed from the circuit board. These chips often use Ball Grid Array (BGA) packages, in which a grid of solder pads on the underside of the silicon chip are used to connect it to the circuit board.

In order to remove these chips, some flux is added to the top before a heat gun is used to melt the solder on the underside of the chip. Once melted, the chip can be removed with tweezers before the solder cools down and re-solidifies, as demonstrated in Figure 1.4.





Once the BGA chip has been removed, any remaining solder on the underside of the array can be cleaned off using a solder wick. This step is important as the tolerances inside the adapter socket are quite small. Adapters can come in various sizes and prices, however they are commonly a clamshell style socket the same size as the chip that is being read, and inside the socket are very small pins that align with necessary pinouts on the ball grid array.

When power is applied, the adapter is able to read the chip and its contents (unless otherwise encrypted). The cleaned chip and its adapter can be seen in Figure 1.5.

Figure 1.5 - Cleaned Chip Ready for Adapter



Once the chip is loaded into the adapter (ensuring it is aligned properly with the pins and arrows), the adapter can be plugged into a laptop. Once plugged in, the adapter will be mounted as a block device with the various partitions of the eMMC chip mounted underneath. Using the "dd" command, the user can copy each partition to the local file system for further examination as seen in Figure 1.6.

Figure 1.6 - Mounted Block Devices to dd

ments/EchoFirm\$ lsblk							
NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINTS	
loop0	7:0	0	4K	1	loop	/snap/bare/5	
loop1	7:1	0	59.2M	1	loop	/snap/core20/1977	
loop2	7:2	0	69.1M	1	loop	/snap/core22/1035	
loop3	7:3	0	230.8M	1	loop	/snap/firefox/3627	
loop4	7:4	0	475.1M	1	loop	/snap/gnome-42-2204/143	
loop5	7:5	0	91.7M	1	loop	/snap/gtk-common-themes/1535	
loop6	7:6	0	109.6M	1	loop	/snap/lxd/24326	
loop7	7:7	0	46.4M	1	loop	/snap/snapd/19459	
sda	8:0	1	3.6G	0	disk		
—sda1	8:1	1	1M	0	part		
—sda2	8:2	1	1M	0	part		
—sda3	8:3	1	1M	0	part		
—sda4	8:4	1	5M	Θ	part		
—sda5	8:5	1	1M	0	part		
—sda6	8:6	1	5M	0	part		
—sda7	8:7	1	10M	0	part		
—sda8	8:8	1	512.5K	Θ	part		
—sda9	8:9	1	16M	0	part		
—sda10	8:10	1	16M	0	part		
—sda11	8:11	1	16M	0	part		
—sda12	8:12	1	16M	0	part		
—sda13	8:13	1	768M	0	part	/media/robert/57f8f4bc-abf4-655f-bf67-946fc0f9f25b	
—sda14	8:14	1	768M	0	part	/media/robert/57f8f4bc-abf4-655f-bf67-946fc0f9f25b1	
—sda15	8:15	1	784M	0	part	/media/robert/57f8f4bc-abf4-655f-bf67-946fc0f9f25b2	
—sda16	259:0	1	1.2G	0	part	/media/robert/57f8f4bc-abf4-655f-bf67-946fc0f9f25b3	
sr0	11:0	1	1024M	0	rom		
vda	252:0	0	64G	0	disk		
—vda1	252:1	0	1G	0	part	/boot/efi	
—vda2	252:2	0	2G	Θ	part	/boot	
-vda3	252:3	0	60.9G	0	part		
└─ubuntuvg-ubuntu-	lv 253:0	0	30.5G	Θ	lvm	/var/snap/firefox/common/host-hunspell	
						/	
[sudo] password for robert:							
2048+0 records in							
2048+0 records out							
1048576 bytes (1.0 MB, 1.0 M1B) copied, 0.057533 s, 18.2 MB/s							
me	ents/EchoFirm	\$					

Figure 1.7 - Copied Partitions

1		cho	Firm\$ ls												
sda10.bin	sda12.bin	sda14.bi	n sda16.bi	In sda2.bi	n sda4.b	oin sda6.b	oin sd	a8.bin							
sda11.bin	sda13.bin	sda15.bi	n sda1.bir	n sda3.bi	n sda5.b	oin sda7.b	oin sd	a9.bin							
1		icho	Firm\$ file												
sda10.bin:	Android boo	otimg, ke	rnel (0x400)80000), ra	ndisk (0×	(44000000),	page	size: 20	048, cmd	iline (bootopt	=64S3,32N2	2,64N2)		
sda11.bin:	Android boo	otimg, ke	rnel (0x400	080000), ra	ndisk (0×	(44000000),	page	size: 20	048, cmd	iline (bootopt	=64S3,32N2	2,64N2)		
sda12.bin:	Android boo	otimg, ke	rnel (0x400	080000), ra	ndisk (0×	(44000000),	page	size: 20	048, cmd	iline (bootopt	=64S3,32N2	2,64N2)		
sda13.bin:	Linux rev 1	1.0 ext4	filesystem	data, UUID	=57f8f4bc	-abf4-6551	f-bf67-	946fc0f9	9f25b (n	needs j	journal	recovery)	(extents)	(large	files)
sda14.bin:	Linux rev 1	1.0 ext4	filesystem	data, UUID	=57f8f4bc	-abf4-6551	f-bf67-	946fc0f9	9f25b (n	needs j	journal	recovery)	(extents)	(large	files)
sda15.bin:	Linux rev 1	1.0 ext4	filesystem	data, UUID	=57f8f4bc	-abf4-6551	f-bf67-	946fc0f9	9f25b (n	needs j	journal	recovery)	(extents)	(large	files)
sda16.bin:	Linux rev 1	1.0 ext4	filesystem	data, UUID	=57f8f4bc	-abf4-6551	f-bf67-	946fc0f9	9f25b (n	needs j	journal	recovery)	(extents)	(large	files)
sda1.bin:	data														
sda2.bin:	data				l										
sda3.bin:	data														
sda4.bin:	data														
sda5.bin:	data														
sda6.bin:	data														
sda7.bin:	data														
sda8.bin:	data														
sda9.bin:	data														
1		icho	Firm\$												

Figure 1.7 shows what types of files each of the copied partitions are. In this case, it would appear that there are three (3) identical Android Boot Images, three (4) identical Linux filesystems, one (1) larger Linux file system, and nine (9) data files. The most likely reason for the various identical partitions could be for ensuring continuity during updates, hot-swapping partitions during update cycles.

Examination will start with the larger Linux filesystem, which can be mounted to the local filesystem using the "mount" command. In doing so, the ext4 filesystem will be expanded, and the user is able to inspect its files and folders directly.

Figure 1.7 - Mounting FS

-									
1		EchoFirm\$ mkdir LargerFileSys							
1		EchoFirm\$ sudo mount sda16.bin LargerFileSys							
1		EchoFirm\$ ls							
LargerFileSys	sda11.bin	sda13.bin	sda15.bin sda1	.bin sda3.bin sda	a5.bin sda7	.bin sda9.b:	in		
sda10.bin	sda12.bin	sda14.bin	sda16.bin sda2	.bin sda4.bin sda	a6.bin sda8	.bin			
robert@hubble	e:~/Documents/	/EchoFirm\$ c	d LargerFileSys,	/					
robert@hubble	e:~/Documents/	/EchoFirm/La	rgerFileSys\$ ls						
acdapi		autotrace	crashreport	dontpanic	local	metrics	property	security	vp
adb	app	bootchart	dalvik-cache	drm	logd	misc	radio	system	
agps_supl	app-asec	<pre>@btmtk</pre>	data	enterprise_certs	lost+found	nfc_socket	resource-cache	tombstones	
alexahybrid	app-lib	bugreports	davs	gps_mnl	media	nvram	securedStorageLocation	user	
amit		cmbd	debug_service	key_provisioning	mediadrm	playready	securestop	vitals	
1		EchoFirm/La	rgerFileSys\$						

Within the expanded filesystem, there are many different files. Since this eMMC chip is effectively the flash memory of the device, data can and will be written to the filesystem upon normal use of the device, meaning there may be some sensitive information included in various locations.

There are numerous ways of searching for this data, from grepping for things like "password" to manually inspecting each file recursively. Many of the files that exist are sqlite3 database files, meaning we can use sqlite to comb through the various databases as seen in Figure 1.8.

Figure 1.8 - Looking Through Sqlite Tables



After doing some digging, the below pieces of sensitive information were identified along with their corresponding file paths. Although this information is somewhat sensitive in nature, the true "crown jewels" of the Amazon Echo were not found, which might be things like the word for word commands that were issued to the device throughout its lifetime, i.e. its search history.

It was noted however, that there was configuration that suggested the use of amazon owned APIs, to which the Echo device likely offloads these "crown jewels" for storage, compute, and data sensitivity reasons.

T ''		a	D	T 1.	C
Figure	1.x -	Sensitive	Data	Identi	ħеd

Type of Data	Filepath						
WiFi network name	/data/com.android.providers.settings/databases/settings.db						
Amazon account owner first & last name	/system/users/0/accounts.db						
Last time the user was active	/system/usagestats/0/yearly/x						
Device's "friendly name"	/local/smarthomed/x5cecx15-cca5-52x0-ax11-6xb040eed3xxx/schema.json						
Nearby bluetooth device names and addresses	/local/whad/btdevice.db.json						
Out-of-box-experience (oobe) web setup private key	/local/oobe-web-setup.cert						
Wifi network name + password	/misc/wifi/wpa_supplicant.conf						
Connected device name, MAC address, and IP address	/misc/dhcp/dnsmasq.leases						

Outlook

This exercise could be likened to a forensic investigation, as the files and folders contained within the firmware of our device were able to be inspected. Though not fully critical in nature, one could begin to imagine how the data extracted from this eMMC package could be leveraged, such as by obtaining the password of a WiFi network for further exploitation.

Additionally, this exercise exemplifies the various ways in which the data that most users never even think about might be stored, captured, and examined. eMMC chips are a unique storage component, however they have become ubiquitous, particularly in low power environments, making it more relevant than ever to understand how to examine these media containers.