



Download Malware? No, thanks.

How Formal Methods can Block Update Attacks

Francesco Mercaldo, Vittoria Nardone, Antonella Santone,
Corrado Aaron Visaggio

Dept. of Engineering - University of Sannio, Benevento, Italy
{fmercaldo, vnardone, santone, visaggio}@unisannio.it



15 May, Austin, Texas, USA

The team

Security Guys

- ▶ Corrado Aaron Visaggio



- ▶ Francesco Mercaldo



Formal Methods Girls

- ▶ Antonella Santone



- ▶ Vittoria Nardone



outline

1. **Motivation and aim**
2. **Proposed Solution**
3. **Evaluation**
4. **Concluding Remarks**



The Problem and the proposed Solution

- Worldwide sales of mobile phones totaled 301 million units in the third quarter of 2014 (Gartner, 2014)
- Existing solutions for protecting privacy and security on smartphones are **still ineffective** in many facets (Marforio et al., 2011, Fedler et al., 2014)
- Easy code transformations make Av-detected Malware unrecognisable (Mercaldo and Visaggio, 2015)
- *Update attack: after an apparently innocuous application is installed on the victim's device, the user is asked to update the application, and a malicious behavior is added to the application.*

Proposed Solution: We investigate about the effectiveness of model checking to discriminate update attacks in Android malware.

Assumption: malware writers usually exploit certain code patterns to realize the update attack.



Motivation

- ▶ A previous work of ours demonstrated that the current mechanism of signature based detection is usually ineffective in detecting malware *
- ▶ trivial code transformations alter the signature of malware, preventing antimalware to detect transformed malware

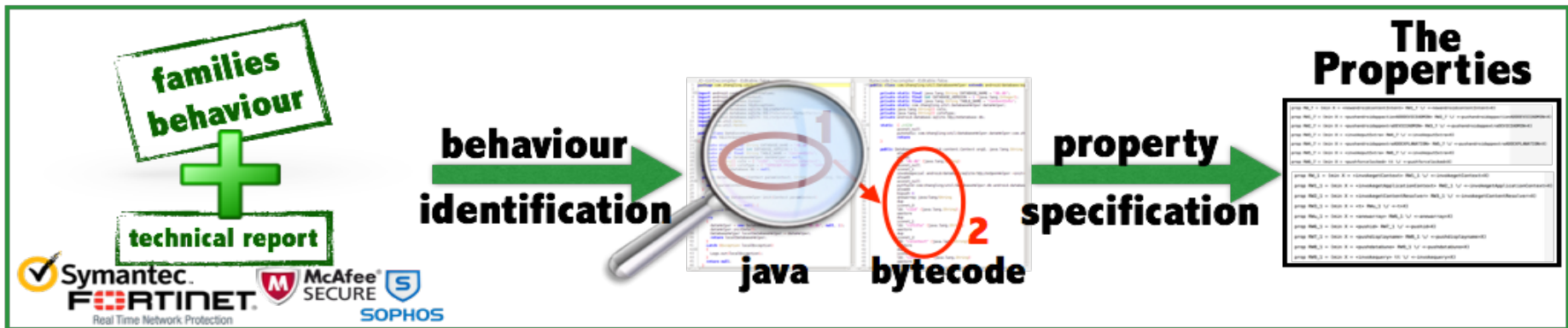
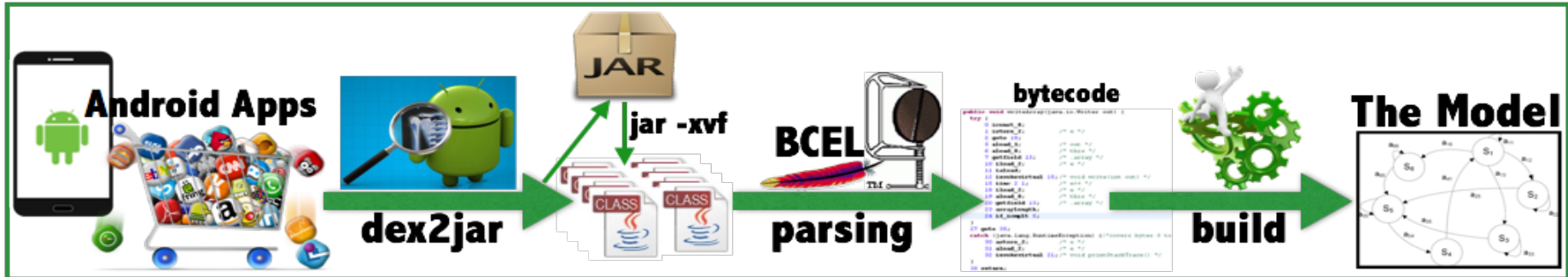
Research Question

RQ: is it possible to identify the update attack by using formal methods ?

*Gerardo Canfora, Andrea Di Sorbo, Francesco Mercaldo, Corrado Aaron Visaggio, *Obfuscation techniques against signature-based detection: a case study*, in Proceedings of Workshop on Mobile System Technologies, IEEE (2015)



The methodology



The method

- ▶ We use the mu-calculus logic as a branching temporal logic to express the behavioural properties
- ▶ We use the Milner's Calculus of Communicating System (CCS) as process algebra
 1. We describe apk through CCS (java bytecode->CCS)
 2. The transform operator is defined for each instruction of the java bytecode
 3. Every instruction that is not a jump is represented by a process including the next instruction
 4. Conditional jump are specified as non-deterministic choice
 5. Unconditional jumps are CCS processes that invoke the corresponding target of the jump



The analyzed behaviour

Update Attack families	Malicious behaviour
Plankton	<ul style="list-style-type: none">✓ Plankton payload is able to silently forward information about the device to a remote location.✓ Once installed on a device, Plankton will send details like IMEI, browser history, permissions granted.✓ The downloaded Dex code launches a connection to the Command server and listens for commands to execute.
AnserverBot	<ul style="list-style-type: none">✓ AnserverBot malicious behavior is embedded into the host app at installation time, i.e., the malicious payload is not downloaded from a remote location but it is stored in an external folder at installation time. Indeed under the raw and the assets directory there are two hidden apps with names anservera.db and anserverb.db.✓ It can make phone calls and download and upgrade the anserverb.db, while anserverb.db is able to update itself and it can independently talk to Command & Control (C&C) servers to fetch and execute subsequent commands.
BaseBridge	<ul style="list-style-type: none">✓ BaseBridge family presents anserverb.db as payload embedded in an external folder like AnserverBot family.✓ The BaseBridge payload is able to receive premium numbers from remote C&C servers and dial calls or send out SMS messages to them, incurring fees for users.
DroidKungFuUpdate	<ul style="list-style-type: none">✓ DroidKungFuUpdate uses the update attack to download the actual payload. But instead of carrying or enclosing the “updated” version inside the original app, it chooses to remotely download a new version from the Web.



From Family Behaviour to Logic Rules (Plankton)

```

ACTIVATION = new Commands("ACTIVATION", 1, "Activation", "Tg8HBgYfBVom");
HOMEPAGE = new Commands("HOMEPAGE", 2, "Homepage", "TgYLHwoZBUkM");
COMMANDS_STATUS = new Commands("COMMANDS_STATUS", 3, "CommandsStatus", "Tg0LHwIICk0aGhURGwc=");
BOOKMARKS = new Commands("BOOKMARKS", 4, "Bookmarks", "TgwLHQEBVwCHQ==");
SHORTCUTS = new Commands("SHORTCUTS", 5, "Shortcuts", "ThOMHR0dB1sdHQ==");
NOTIFICATIONS = new Commands("NOTIFICATIONS", 6, "Notifications", "TgALBgYPDU0IGhOKAAc=");
TERMINATE = new Commands("TERMINATE", 7, "Terminate", "ThoBAAIACK8dCw==");
DUMP_LOG = new Commands("DUMP_LOG", 8, "DumpLog", "TgoRHx8FC0k=");
UNEXPECTED_EXCEPTION = new Commands("UNEXPECTED_EXCEPTION", 9, "UnexpectedException", "ThsKFxcZAU0dCxAAFhdLEgYGB0");
UPGRADE = new Commands("UPGRADE", 10, "Upgrade", "ThsUFR0IAEs=");
INSTALLATION = new Commands("INSTALLATION", 11, "Installation", "TgcKARsICEIIGhOKAA=");
INFO = new Commands("INFO", 12, "Info", "TgcKFAA=");
OPTOUT = new Commands("OPTOUT", 13, "Optout", "TgEUBgAcEA==");
Commands[] arrayOfCommands = new Commands[14];
    
```

1

```

new com/apperhand/common/dto/Commands$Commands
dup
ldc "COMMANDS" (java.lang.String)
iconst_0
ldc "Commands" (java.lang.String)
ldc "Tg8HBgYfBVom" (java.lang.String)
invokespecial com/apperhand/common/dto/Commands$Commands <init>((Ljava/lang/String;ILjava/lang/String;Ljava/lang/String;)V)
putstatic com/apperhand/common/dto/Commands$Commands.COMMANDS:com.apperhand.common.dto.Commands$Commands
new com/apperhand/common/dto/Commands$Commands
dup
ldc "ACTIVATION" (java.lang.String)
iconst_1
ldc "Activation" (java.lang.String)
ldc "Tg8HBgYfBVom" (java.lang.String)
invokespecial com/apperhand/common/dto/Commands$Commands <init>((Ljava/lang/String;ILjava/lang/String;Ljava/lang/String;)V)
putstatic com/apperhand/common/dto/Commands$Commands.ACTIVATION:com.apperhand.common.dto.Commands$Commands
new com/apperhand/common/dto/Commands$Commands
dup
ldc "HOMEPAGE" (java.lang.String)
iconst_2
ldc "Homepage" (java.lang.String)
ldc "TgYLHwoZBUkM" (java.lang.String)
invokespecial com/apperhand/common/dto/Commands$Commands <init>((Ljava/lang/String;ILjava/lang/String;Ljava/lang/String;)V)
putstatic com/apperhand/common/dto/Commands$Commands.HOMEPAGE:com.apperhand.common.dto.Commands$Commands
new com/apperhand/common/dto/Commands$Commands
dup
ldc "COMMANDS_STATUS" (java.lang.String)
iconst_3
ldc "CommandsStatus" (java.lang.String)
ldc "Tg0LHwIICk0aGhURGwc=" (java.lang.String)
invokespecial com/apperhand/common/dto/Commands$Commands <init>((Ljava/lang/String;ILjava/lang/String;Ljava/lang/String;)V)
putstatic com/apperhand/common/dto/Commands$Commands.COMMANDS_STATUS:com.apperhand.common.dto.Commands$Commands
new com/apperhand/common/dto/Commands$Commands
dup
ldc "BOOKMARKS" (java.lang.String)
iconst_4
ldc "Bookmarks" (java.lang.String)
ldc "TgwLHQEBVwCHQ==" (java.lang.String)
invokespecial com/apperhand/common/dto/Commands$Commands <init>((Ljava/lang/String;ILjava/lang/String;Ljava/lang/String;)V)
putstatic com/apperhand/common/dto/Commands$Commands.BOOKMARKS:com.apperhand.common.dto.Commands$Commands
    
```

2-A

```

new com/apperhand/common/dto/Command$Commands
dup
ldc "ACTIVATION" (java.lang.String)
iconst_1
ldc "Activation" (java.lang.String)
ldc "Tg8HBgYfBVom" (java.lang.String)
invokespecial com/apperhand/common/dto/Command$Commands <init>((Ljava/lang/String;ILjava/lang/String;Ljava/lang/String;)V)
putstatic com/apperhand/common/dto/Command$Commands.ACTIVATION:com.apperhand.common.dto.Command$Commands
new com/apperhand/common/dto/Command$Commands
dup
ldc "HOMEPAGE" (java.lang.String)
iconst_2
ldc "Homepage" (java.lang.String)
ldc "TgYLHwoZBUkM" (java.lang.String)
invokespecial com/apperhand/common/dto/Command$Commands <init>((Ljava/lang/String;ILjava/lang/String;Ljava/lang/String;)V)
putstatic com/apperhand/common/dto/Command$Commands.HOMEPAGE:com.apperhand.common.dto.Command$Commands
    
```

2-B

```

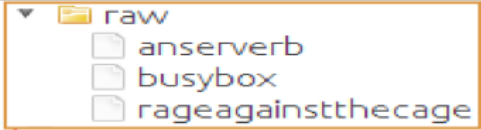
prop RW5_0 = (min X = <pushACTIVATION> RW6_0 \\/ <-pushACTIVATION>X)
prop RW6_0 = (min X = <pushActivation> RW7_0 \\/ <-pushActivation>X)
prop RW7_0 = (min X = <pushTgOottoHBgYfBVom> RW8_0 \\/ <-pushTgOottoHBgYfBVom>X)
    
```

3



From Family Behaviour to Logic Rules (AnserverBot)

```
c.aSpark(2130968577, "busybox", localContext);
c.bSpark(2130968576, "SMSApp.apk", localContext);
String str = getFilesDir().getAbsolutePath();
c.aSpark(localFileOutputStream, "chmod 777 " + str + "/busybox");
c.aSpark(localFileOutputStream, "chmod 777 " + str + "/" + "SMSApp.apk");
c.aSpark(localFileOutputStream, str + "/busybox mount -o remount,rw /system");
c.aSpark(localFileOutputStream, str + "/busybox cp -rp " + str + "/" + "SMSApp.apk" + " /system/app/");
c.aSpark(localFileOutputStream, "pm install -r " + str + "/" + "SMSApp.apk");
c.aSpark(localFileOutputStream, str + "/busybox chown 0 /system/app/" + "SMSApp.apk");
c.aSpark(localFileOutputStream, str + "/busybox killall rageagainstthecage");
c.aSpark(localFileOutputStream, "checkvar=checked");
c.aSpark(localFileOutputStream, "echo finished $checkvar");
```

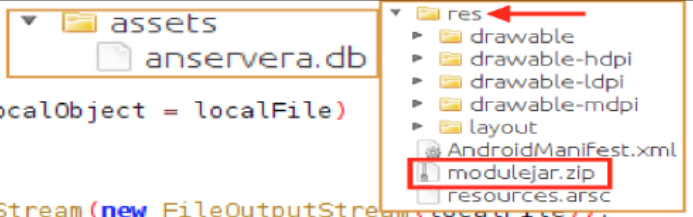


A-1

```
prop BB_2 = (min X = <pushrawbusybox,pushbusybox> BB1_2 \\/ <-pushrawbusybox,pushbusybox>X)
prop BB1_2= (min X = <pushrawanserverb,pushSMSAppapk> BB2_2 \\/ <-pushrawanserverb,pushSMSAppapk>X)
prop BB2_2= (min X = <pushSMSAppapk,invokegetFilesDir> BB3_2 \\/ <-pushSMSAppapk,invokegetFilesDir>X) ...
prop BB7_2= (min X = <pushbusyboxmountoremounttrwsystem> BB8_2 \\/ <-pushbusyboxmountoremounttrwsystem>X) ...
prop BB11_2= (min X = <pushpminstallr> BB12_2 \\/ <-pushpminstallr>X) ...
prop BB15_2= (min X = <pushbusyboxkillallrageagainstthecage> tt \\/ <-pushbusyboxkillallrageagainstthecage>X)
```

A-2

```
ZipFile localZipFile = new ZipFile(paramFile);
Enumeration localEnumeration = localZipFile.entries();
byte[] arrayOfByte = new byte['E'];
File localFile;
for (Object localObject = null; localEnumeration.hasMoreElements(); localObject = localFile)
{
    ZipEntry localZipEntry = (ZipEntry)localEnumeration.nextElement();
    localFile = new File(paramFile.getAbsolutePath().concat(".apk"));
    BufferedOutputStream localBufferedOutputStream = new BufferedOutputStream(new FileOutputStream(localFile));
    BufferedInputStream localBufferedInputStream = new BufferedInputStream(localZipFile.getInputStream(localZipEntry));
```



B-1

```
prop ANSB_1 = (min X = <newjavautlizipZipFile> ANSB1_1 \\/ <-newjavautlizipZipFile>X)
prop ANSB1_1= (min X = <invokeentries> ANSB2_1 \\/ <-invokeentries>X)
prop ANSB2_1= (min X = <checkcastjavautlizipZipEntry> ANSB3_1 \\/ <-checkcastjavautlizipZipEntry>X)
prop ANSB3_1= (min X = <invokegetAbsolutePath> ANSB4_1 \\/ <-invokegetAbsolutePath>X)
prop ANSB4_1= (min X = <pushapk> ANSB5_1 \\/ <-pushapk>X)
prop ANSB5_1= (min X = <invokeconcat> tt \\/ <-invokeconcat>X)
```

B-2



The malware dataset

- ▶ Malware + Trusted
- ▶ Compare our method with the top 10 antimalware (according to AV.Test)
- ▶ We verified our method and the 10 antimalware on obfuscated code
- ▶ We developed a framework able to inject several obfuscation levels:
 - ▶ (i) changing package name;
 - ▶ (ii) identifier renaming;
 - ▶ (iii) data encoding;
 - ▶ (iv) call indirections;
 - ▶ (v) core reordering;
 - ▶ (vi) junk code insertion.

Full
dataset:
#2,581
samples

Real-world malware
belonging to Drebin
and Genome
Project

Dataset	Original Samples	Morphed Samples	#Samples for Family
Plankton	625	543	1,168
BaseBridge	330	307	637
AnserverBot	187	187	374
DroidKungFuUpdate	1	1	2
Repackaged Attack	200	0	200
Trusted	200	0	200
Total	1,543	1,038	2,581



Antimalware against Our Method



Antimalware	Plankton			AnserverBot			BaseBridge			DroidKungFuUpdate		
	%ident.	#ident.	#unident.	%ident.	#ident.	#unident.	%ident.	#ident.	#unident.	%ident.	#ident.	#unident.
Original												
<i>AhnLab</i>	8.16%	51	574	93.05%	174	13	29.09%	96	234	100%	1	0
<i>Alibaba</i>	0%	0	625	0%	0	187	0%	0	330	0%	0	1
<i>Antiy</i>	0%	0	625	6.42%	12	175	0%	0	330	0%	0	1
<i>Avast</i>	2.8%	13	612	5.35%	10	177	33.03%	109	221	%	0	1
<i>AVG</i>	4%	25	600	0%	0	187	0%	0	330	0%	0	1
<i>Avira</i>	79.68%	498	127	0%	0	187	14.55%	48	282	100%	1	0
<i>Baidu</i>	59.2%	370	255	82.89%	155	32	32.12%	106	224	0%	0	1
<i>BitDefender</i>	73.36%	496	129	0%	0	187	91.52%	302	28	0%	0	1
<i>ESET-NOD32</i>	96%	600	25	0%	0	187	36.36%	120	210	100%	1	0
<i>GData</i>	79.2%	495	130	0%	0	187	91.52%	302	28	0%	0	1
<i>Our Method</i>	100%	625	0	100%	330	0	98.79%	326	4	100%	1	0
Morphed												
<i>AhnLab</i>	0.74%	4	539	73.8%	138	49	11.08%	34	273	0%	0	1
<i>Alibaba</i>	0%	0	543	0%	0	187	0%	0	307	0%	0	1
<i>Antiy</i>	0%	0	543	1.6%	3	184	0%	0	307	0%	0	1
<i>Avast</i>	1.29%	7	536	1.07%	2	185	15.31%	47	260	0%	0	1
<i>AVG</i>	2.58%	14	529	0%	0	187	0%	0	307	0%	0	1
<i>Avira</i>	34.62%	188	355	0%	0	187	4.56%	14	293	100%	1	0
<i>Baidu</i>	27.26%	148	395	68.45%	128	59	7.49%	23	284	0%	0	1
<i>BitDefender</i>	26.89%	146	397	0%	0	187	28.99%	89	218	0%	0	1
<i>ESET-NOD32</i>	36.28%	197	346	0%	0	187	6.84%	21	286	100%	1	0
<i>GData</i>	38.31%	208	335	0%	0	187	37.13%	114	193	%	0	1
<i>Our Method</i>	100%	543	0	100%	187	0	98.7%	303	4	100%	1	0



Performance Evaluation

	#samples	TP	FP	FN	TN	PR	RC	Fm	Acc
Plankton	1,168	1,168	0	0	1,413	1	1	1	1
BaseBridge	637	629	0	8	1,944	1	0.98	0.98	0.99
AnserverBot	374	374	0	0	2,207	1	1	1	1
DroidKungFu Update	2	2	0	0	2,579	1	1	1	1

$$PR = \frac{TP}{TP+FP}; RC = \frac{TP}{TP+FN};$$
$$Fm = \frac{2PR RC}{PR+RC}; Acc = \frac{TP+TN}{TP+FN+FP+TN}$$



Rq Response

- ▶ RQ: our method is promising to identify update attack malware payload.
 - ▶ the gap between our approach and the signature-based detection is broader in the morphed sample evaluation
 - ▶ we outperform the top 10 current signature-based approaches in detecting morphed samples
 - ▶ our method is transparent with respect to obfuscation



Remarks and future works

- ▶ We use model checking in order to test our model against one of most diffused attack in Android environment: the update attack
- ▶ We obtain:
 - ▶ 100% accuracy in Plankton family identification;
 - ▶ 99% accuracy in BaseBridge family identification;
 - ▶ 100% Accuracy in AnserverBot family identification;
 - ▶ 100% Accuracy in DroidKungFuUpdate family identification.
- ▶ As future work we are going to extend our preliminary evaluation to other widespread families.
- ▶ In addition we plan to track the phylogenesis of malware to characterize the payload family tree



Thanks for your attention

- ▶ We are grateful for receiving comments, observations, suggestions, and collaborations with other research groups which could improve our research.

