SmartGen: Exposing Server URLs of Mobile Apps with Selective Symbolic Execution

Chaoshun Zuo Zhiqiang Lin

Department of Computer Science
University of Texas at Dallas

April 6th, 2017

Server URLs

`https://www.google.com/search?q=www+2017`

Server URLs

`https://www.google.com/search?q=www+2017`

A URL includes

1. Domain name
2. Resource path
3. Query parameters
4. ...

Server URLs

`https://www.google.com/search?q=www+2017`

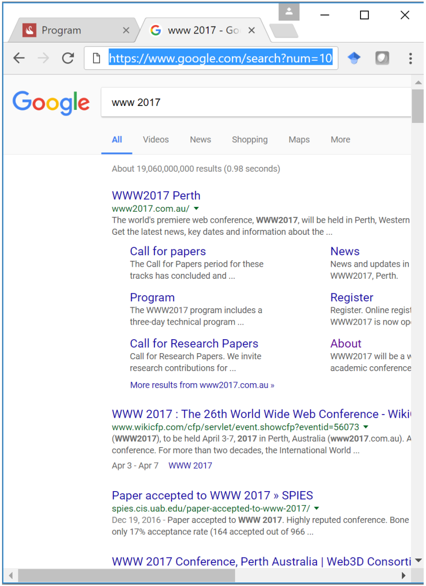
A URL includes

1. Domain name
2. Resource path
3. Query parameters
4. ...

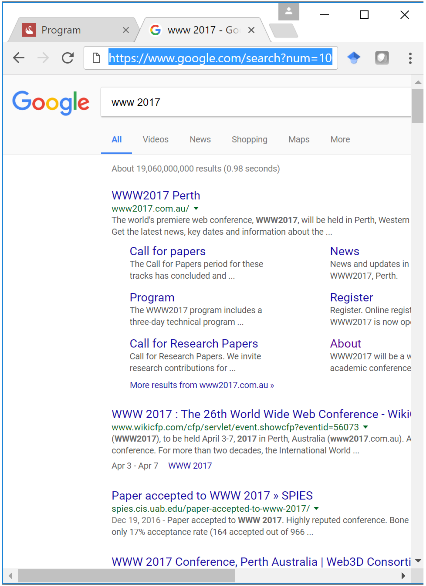
Security Applications

1. Hidden service identification
2. Malicious website detection
3. Server vulnerability fuzzing
4. ...

Browsers' URLs vs. Mobile Apps' URLs



Browsers' URLs vs. Mobile Apps' URLs



Source: `cloudxtension.com`

Security Implications of the URLs in Mobile Apps

Source: cloudxtension.com

1. Hiding the URLs may allow the servers to collect some **private sensitive information**
2. Mobile apps may talk to some **unwanted services** (e.g., malicious ads sites)
3. **False illusions** (security through obscurity) to the app developers that their services are secure (server URLs are hidden, none knows and none will attack (or fuzz) them).

Security Implications of the URLs in Mobile Apps

Source: `cloudxtension.com`

1. Hiding the URLs may allow the servers to collect some **private sensitive information**
2. Mobile apps may talk to some **unwanted services** (e.g., malicious ads sites)
3. **False illusions** (security through obscurity) to the app developers that their services are secure (server URLs are hidden, none knows and none will attack (or fuzz) them).

It is imperative to **expose the server URLs from mobile apps**

A Movitating Example: ShopClues

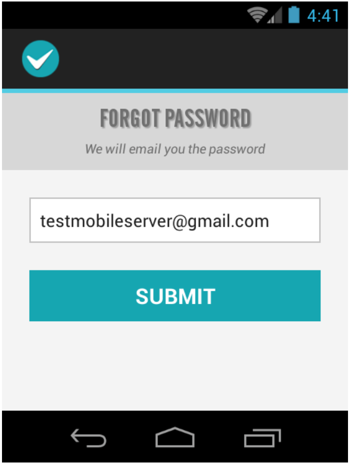
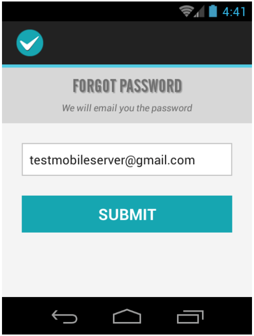


Figure: The password reset activity of ShopClues (between 10 million and 50 million installs).

A Movitating Example: ShopClues




```
PUT /api/v9/forgotpassword?key=d12121c70dda5edfgd1df6633fdb36c0
HTTP/1.1
Content-Type: application/json
Connection: close
User-Agent: Dalvik/1.6.0 (Linux; Android 4.2)
Host: sm.shopclues.com
Accept-Encoding: gzip
Content-Length: 73

{"user_email":"testmobileserver@gmail.com","key":"d12121c70dda5e
dfgd1df6633fdb36c0"}
```

Which Analysis We Should Use?

Static Analysis vs. Dynamic Analysis vs. Symbolic Execution

```
PUT /api/v9/forgotpassword?key=d12121c70dda5edfgd1df6633fdb36c0
HTTP/1.1
Content-Type: application/json
Connection: close
User-Agent: Dalvik/1.6.0 (Linux; Android 4.2)
Host: sm.shopclues.com
Accept-Encoding: gzip
Content-Length: 73

{"user_email":"testmobileserver@gmail.com","key":"d12121c70dda5e
dfgd1df6633fdb36c0"}
```

Which Analysis We Should Use?

Static Analysis vs. Dynamic Analysis vs. Symbolic Execution

```
PUT /api/v9/forgotpassword?key=d12121c70dda5edfgd1df6633fdb36c0
HTTP/1.1
Content-Type: application/json
Connection: close
User-Agent: Dalvik/1.6.0 (Linux; Android 4.2)
Host: sm.shopclues.com
Accept-Encoding: gzip
Content-Length: 73

{"user_email":"testmobileserver@gmail.com","key":"d12121c70dda5e
dfgd1df6633fdb36c0"}
```

Static Analysis

- String cantenation
- Crypto keys

Which Analysis We Should Use?

Static Analysis vs. Dynamic Analysis vs. Symbolic Execution

```
PUT /api/v9/forgotpassword?key=d12121c70dda5edfgd1df6633fdb36c0
HTTP/1.1
Content-Type: application/json
Connection: close
User-Agent: Dalvik/1.6.0 (Linux; Android 4.2)
Host: sm.shopclues.com
Accept-Encoding: gzip
Content-Length: 73

{"user_email":"testmobileserver@gmail.com","key":"d12121c70dda5e
dfgd1df6633fdb36c0"}
```

Static Analysis

- String cantenation
- Crypto keys

Dynamic Analysis

- Random inputs
- Incompleteness
- ...

Which Analysis We Should Use?

Static Analysis vs. Dynamic Analysis vs. Symbolic Execution

```
PUT /api/v9/forgotpassword?key=d12121c70dda5edfgd1df6633fdb36c0
HTTP/1.1
Content-Type: application/json
Connection: close
User-Agent: Dalvik/1.6.0 (Linux; Android 4.2)
Host: sm.shopclues.com
Accept-Encoding: gzip
Content-Length: 73

{"user_email":"testmobileserver@gmail.com","key":"d12121c70dda5e
dfgd1df6633fdb36c0"}
```

Static Analysis

- String cantenation
- Crypto keys

Dynamic Analysis

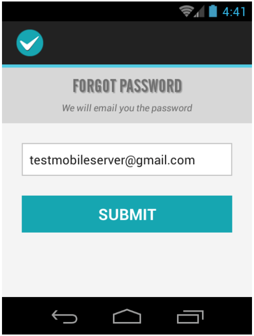
- Random inputs
- Incompleteness
- ...

Symbolic Execution

- Systematic
- Automated
- ...

Symbolic Execution

Generating Inputs Based on Program Code



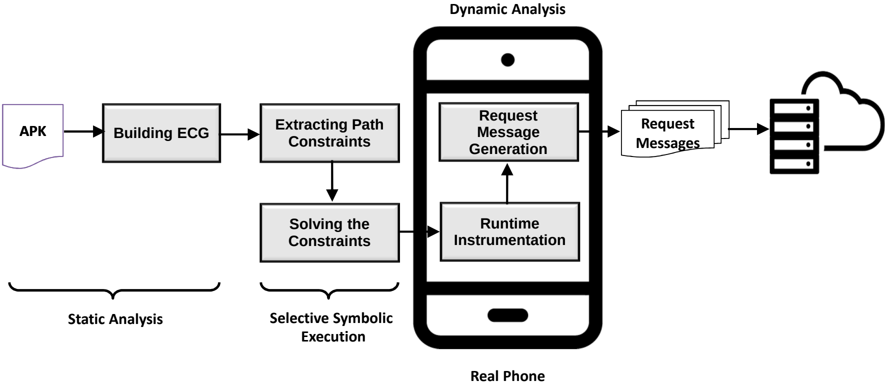

```
package com.shopclues;

class y implements View$OnClickListener {
    EditText b;
    ...
    public void onClick(View arg5) {
        String v0 = this.b.getText().toString().trim();
        if(v0.equalsIgnoreCase("")) {
            Toast.makeText(this.a, "Email Id should not be
            empty", 1).show();
        }
        else if(!al.a(v0)) {
            Toast.makeText(this.a, "The email entered is not
            a valid email", 1).show();
        }
        else if(al.b(this.a)) {
            this.a.c = new ac(this.a, v0);
            this.a.c.execute(new Void[0]);
        }
        else {
            Toast.makeText(this.a, "Please check your
            internet connection", 1).show();
        }
    }
}
```

Various Constraints in Mobile Apps

Various Constraints

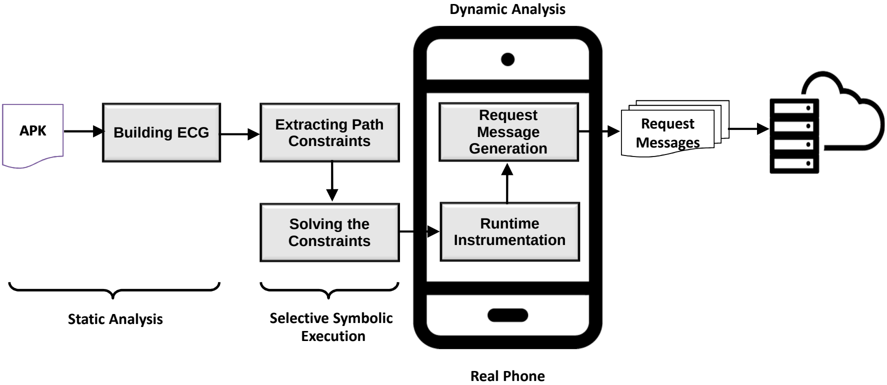
1. Two text-box's inputs need to be equivalent
2. The "age" needs to be greater than 18
3. A "zip code" needs to be a five digit sequence
4. A "phone number" needs to be a phone number
5. A file name extension needs to be some type (e.g., `jpg`)
6. ...

Introducing SmartGen



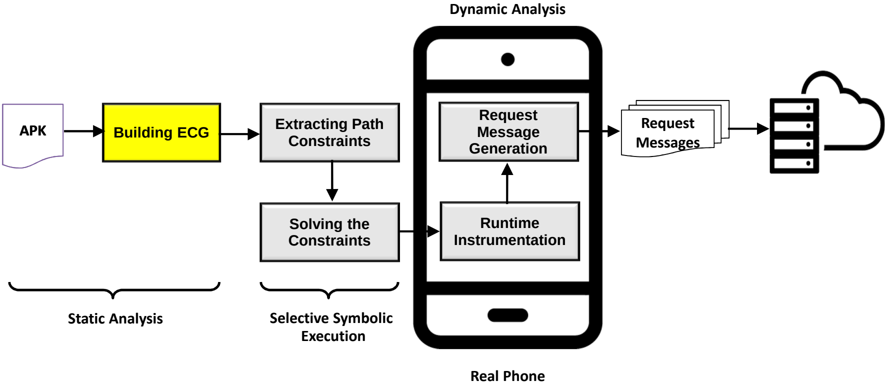
- Automated
- Systematic
- Scalable

Introducing SmartGen



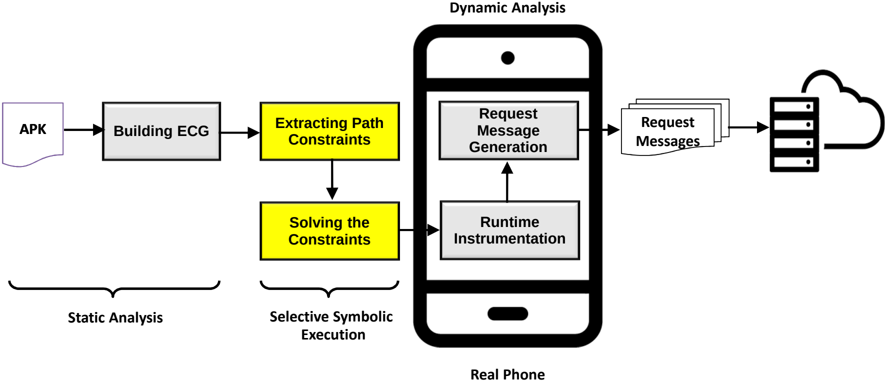
- Static analysis
- Selective symbolic execution
- Dynamic analysis

Static Analysis



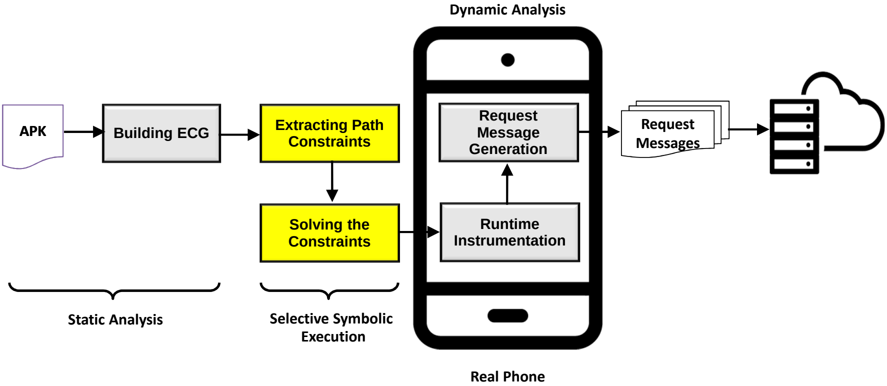
- Using soot [soo] framework
- Building extended call graph (ECG)
- EdgeMiner [CFB+15] for callbacks

Selective Symbolic Execution



- Data flow analysis (w/ FlowDroid [ARF+14])
- Extract the path constraints
- Solve them w/ Z3-str [ZZG13]

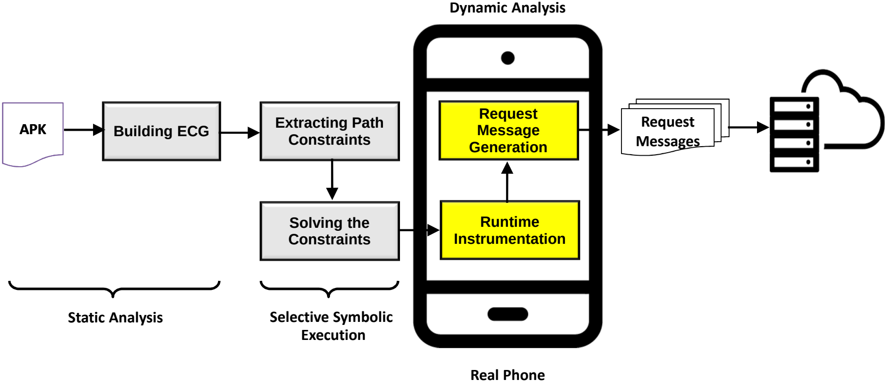
Selective Symbolic Execution



- Data flow analysis (w/ FlowDroid [ARF+14])
- Extract the path constraints
- Solve them w/ Z3-str [ZZG13]

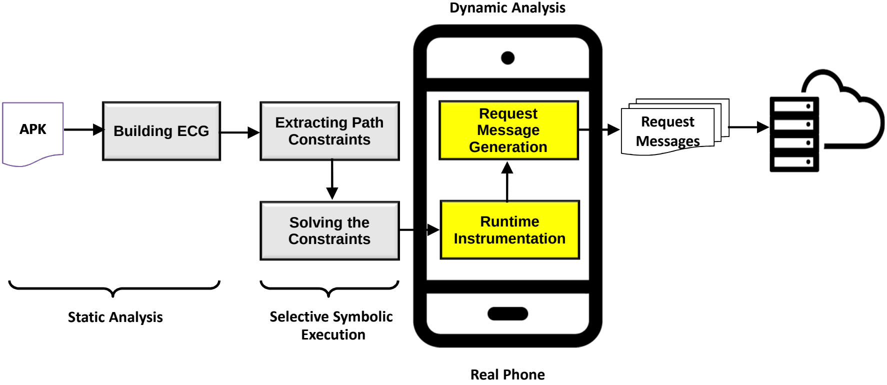
Why Selective: only on the execution path of network sending APIs (to trigger the request messages)

Runtime Instrumentation



- System code static rewriting
- Repackaging the apps
- System debugging tool `adb`

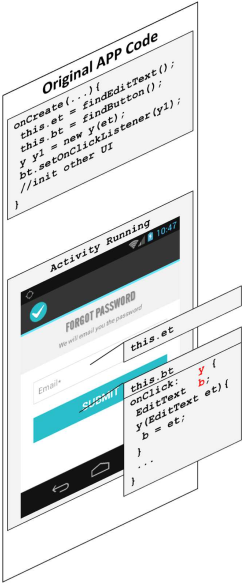
Runtime Instrumentation



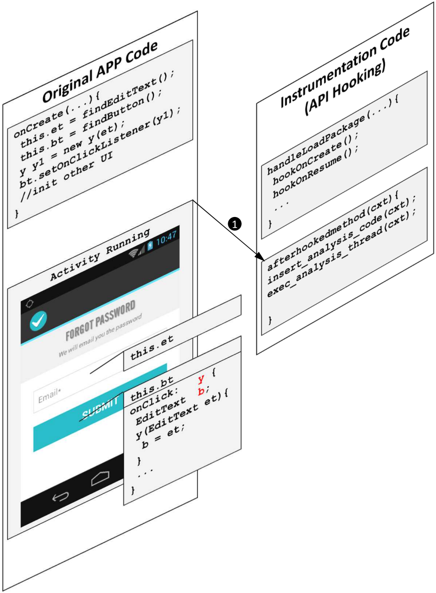
- System code static rewriting
- Repackaging the apps
- System debugging tool `adb`

- A new approach that leverages **API hooking** and **Java reflection**

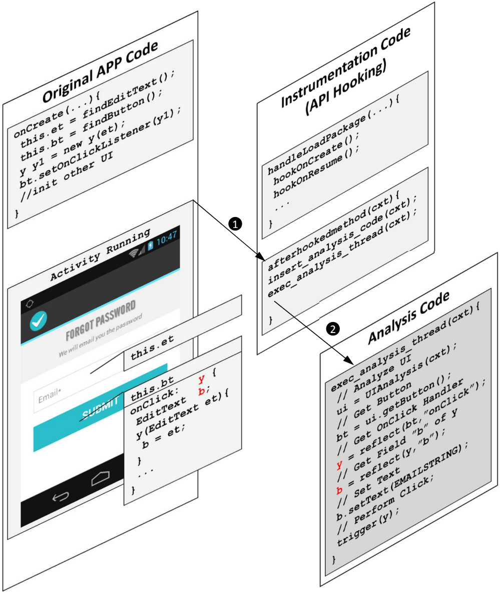
Runtime Instrumentation



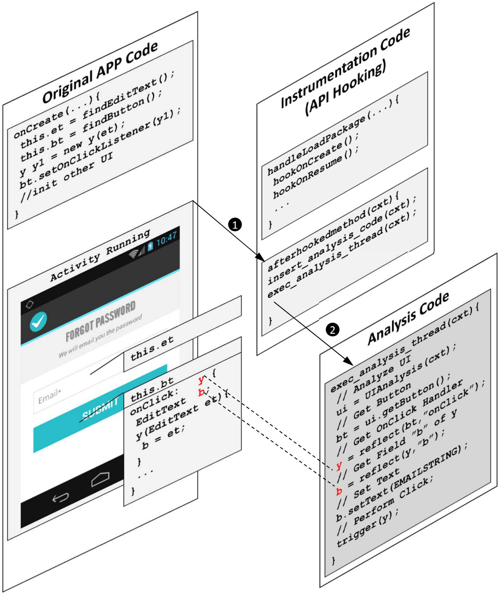
Runtime Instrumentation



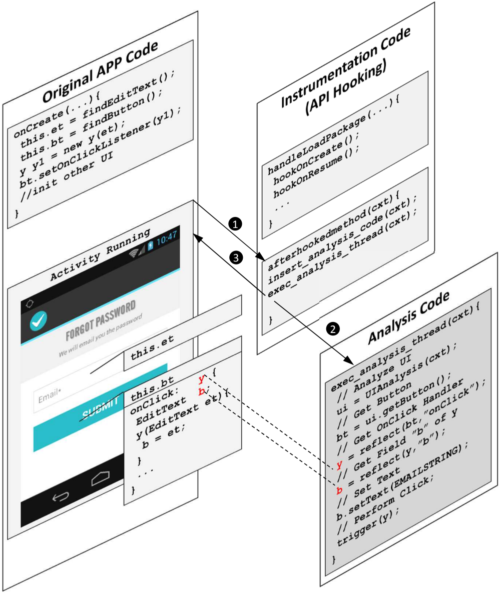
Runtime Instrumentation



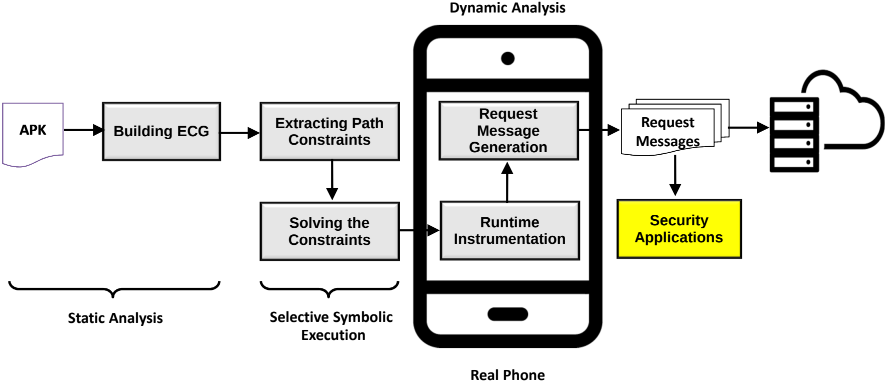
Runtime Instrumentation



Runtime Instrumentation

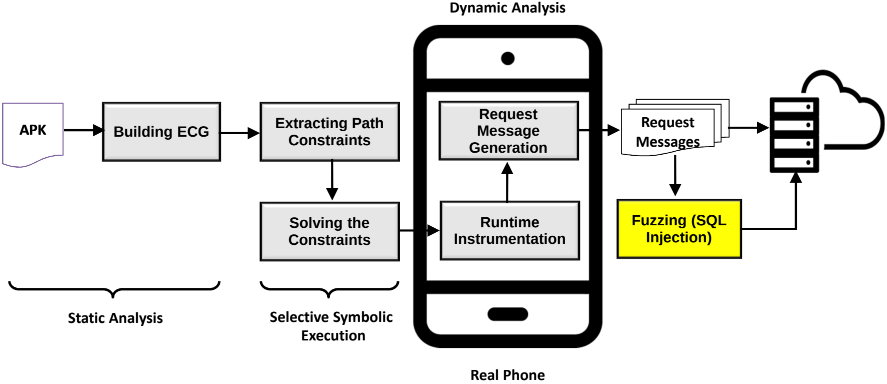


Security Applications



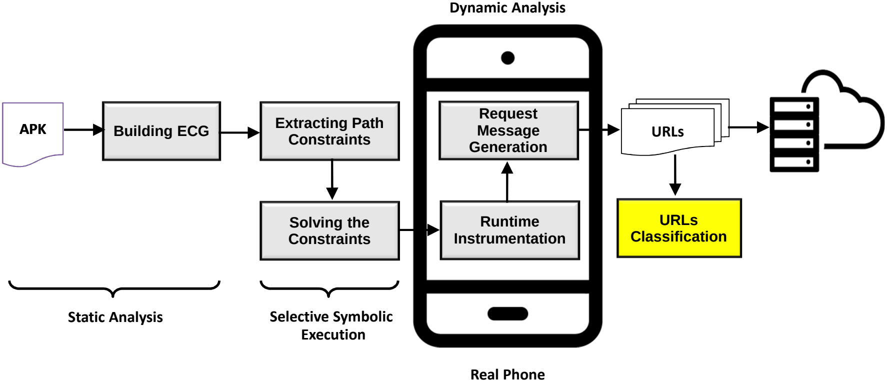
- SQL Injection
- Cross Site Scripting
- Others (e.g., malicious URL detection)

SQL Injection



- “SELECT PG_SLEEP(5);”, “SELECT PG_SLEEP(10);”
- “';WAITFOR DELAY '0:0:5'–”
- “;SELECT COUNT(*) FROM SYSIBM.SYSTABLES”

Malicious URL Detection



- Malware sites
- Compromised sites
- `VirusTotal` provides services for these detections

Overall Experimental Results

Item	Value
# Apps	5,000
Size of the Dataset (G-bytes)	126.2
Time of the first two phases analyses (s)	90,143 (25 hours)
# Targeted API Calls	147,327
# Constraints	47,602
# UI Configuration files generated	25,030
Time of Dynamic Analysis (s)	486,446 (135 hours)
# Request Messages	257,755
# Exposed URLs	297,780
# Unique Domains	18,193
Logged Message Size (G-bytes)	24.0
Σ Malicious URLs	8,634

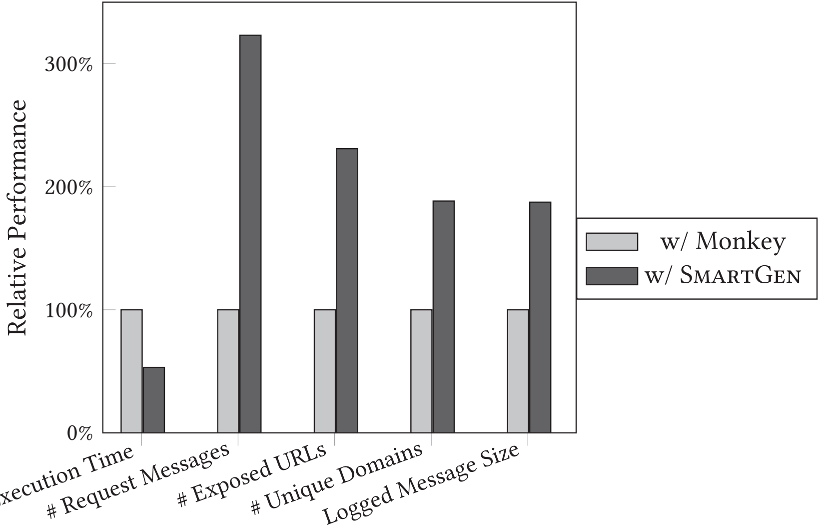
Overall Experimental Results

Item	Value
# Apps	5,000
Size of the Dataset (G-bytes)	126.2
Time of the first two phases analyses (s)	90,143 (25 hours)
# Targeted API Calls	147,327
# Constraints	47,602
# UI Configuration files generated	25,030
Time of Dynamic Analysis (s)	486,446 (135 hours)
# Request Messages	257,755
# Exposed URLs	297,780
# Unique Domains	18,193
Logged Message Size (G-bytes)	24.0
Σ Malicious URLs	8,634

Statistics on the Extracted String Constraints

Constraints Name	# Constraints
Not null	25,855
String_length	13,858
String_isEmpty	377
String_contains	196
String_contentEquals	43
String_equals	3,087
String_equalsIgnoreCase	991
String_matches	448
String_endsWith	11
String_startsWith	64
TextUtils_isEmpty	2,355
Matcher_matches	317

Comparison w/ Monkey [mon]



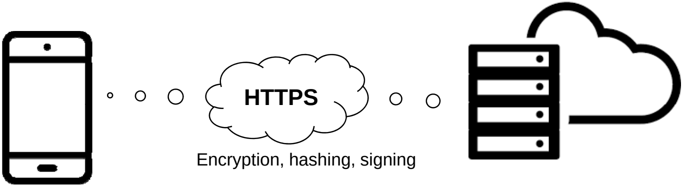
Security Application: Malicious URL detection

Detection Engine	#Phishing Sites	#Malware	#Malicious Sites	Σ #Harmful URLs
ADMINUSLabs	0	0	4	4
AegisLab WebGuard	0	0	1	1
AutoShun	0	0	863	863
Avira	2062	941	0	3003
BitDefender	0	191	0	191
Blueliv	0	0	5	5
CLEAN MX	0	0	14	14
CRDF	0	0	150	150
CloudStat	0	0	1	1
Dr.Web	0	0	2330	2330
ESET	0	75	0	75
Emsisoft	1	43	0	44
Fortinet	8	469	0	477
Google Safebrowsing	0	13	2	15
Kaspersky	0	2	0	2
Malwarebytes hpHosts	0	1103	0	1103
ParetoLogic	0	800	0	800
Quick Heal	0	0	2	2
Quttera	0	0	6	6
SCUMWARE.org	0	8	0	8
Sophos	0	0	56	56
Sucuri SiteCheck	0	0	248	248
ThreatHive	0	0	8	8
Trustwave	0	0	80	80
Websense ThreatSeeker	0	0	56	56
Yandex Safebrowsing	0	173	0	173
Σ#Harmful URLs	2071	3818	3826	9715
Σ#Unique Harmful URLs	2071	3722	3228	8634

Related Work

1. **Dynamic Analysis.** Monkey [mon] automatically executes and randomly navigates an app. AppsPlayground [RCE13] and SMV-Hunter [SSG+14] more intelligent. A3E [AN13], a targeted exploration of mobile apps. DynoDroid [MTN13] instruments the Android framework and uses `adb` to monitor UI interaction and generate UI events.
2. **Symbolic Execution.** Symbolic execution in app testing in general [MMP+12], path exploration [ANHY12], and malware analysis [WL16]. Closely related work **IntelliDroid** but it only focuses on malware and lacks generality of UI rich mobile app analysis.

Related Work

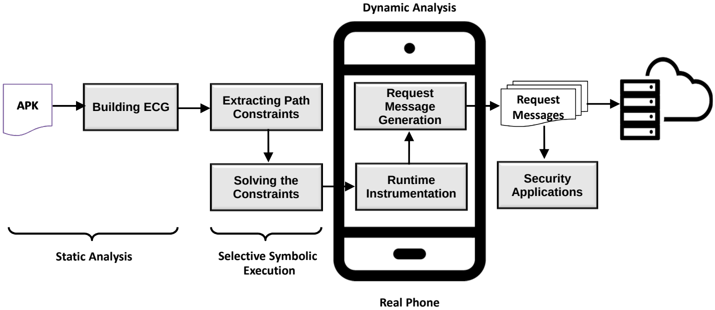


1. **Mobile App Vulnerability Discovery.** A large body of efforts have focused on discovering vulnerabilities in mobile apps. TaintDroid [EGC+10], PiOS [EKKV11], CHEX [LLW+12], SMV-Hunter [SSG+14].

1. **Remote Server Vulnerability Discovery.** Few efforts (e.g., AutoForge [ZWWL16]) including smartgen [ZL17]. have been focusing on identifying the vulnerabilities in **app's server side.**

SMARTGEN [ZL17]

A Fully Automated, Symbolic Execution Based, Mobile App Execution Framework



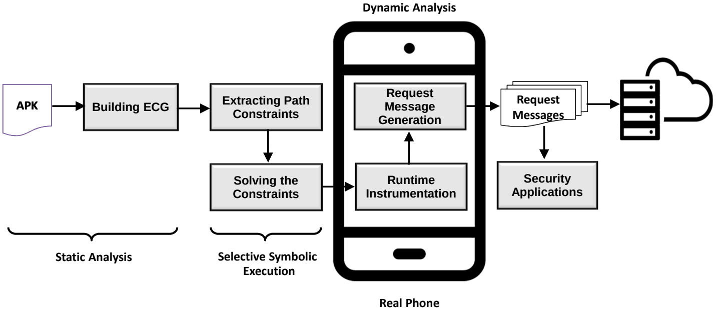
SMARTGEN

- A **fully automated** mobile app execution framework via **symbolic execution**
- Can be used to **test various security vulnerabilities in mobile systems**

Experimental Result w/ 5,000 apps

- Each app has 1,000,000 installs
- These apps actually talk to 2,071 phishing sites, 3,722 malware sites, and 3,228 malicious sites

Thank You



Acknowledgement

- AFOSR, NSF
- VirusTotal (premium services)

Q&A

firstname.lastname@utdallas.edu

References I

- Tanzirul Azim and Iulian Neamtiu, *Targeted and depth-first exploration for systematic testing of android apps*, Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications (New York, NY, USA), OOPSLA '13, ACM, 2013, pp. 641–660.
- Saswat Anand, Mayur Naik, Mary Jean Harrold, and Hongseok Yang, *Automated concolic testing of smartphone apps*, Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (New York, NY, USA), FSE '12, ACM, 2012, pp. 59:1–59:11.
- Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel, *Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps*, Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (New York, NY, USA), PLDI '14, ACM, 2014, pp. 259–269.
- Marshall Beddoe, *The protocol informatics project*, `http://www.4tphi.net/~awalters/PI/PI.html`.
- Yinzhi Cao, Yanick Fratantonio, Antonio Bianchi, Manuel Egele, Christopher Kruegel, Giovanni Vigna, and Yan Chen, *Edgeminer: Automatically detecting implicit control flow transitions through the android framework.*, Proceedings of the 20th Annual Network and Distributed System Security Symposium (NDSS'15), 2015.
- Weidong Cui, Jayanthkumar Kannan, and Helen J. Wang, *Discoverer: Automatic protocol reverse engineering from network traces*, Proceedings of the 16th USENIX Security Symposium (Security'07) (Boston, MA), August 2007.
- Juan Caballero, Pongsin Poosankam, Christian Kreibich, and Dawn Song, *Dispatcher: Enabling active botnet infiltration using automatic protocol reverse-engineering*, Proceedings of the 16th ACM Conference on Computer and and Communications Security (CCS'09) (Chicago, Illinois, USA), 2009, pp. 621–634.

References II

- Weidong Cui, Vern Paxson, Nicholas Weaver, and Randy H. Katz, *Protocol-independent adaptive replay of application dialog*, Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS'06) (San Diego, CA), February 2006.
- Juan Caballero and Dawn Song, *Polyglot: Automatic extraction of protocol format using dynamic binary analysis*, Proceedings of the 14th ACM Conference on Computer and and Communications Security (CCS'07) (Alexandria, Virginia, USA), 2007, pp. 317–329.
- W. Enck, P. Gilbert, B.G. Chun, L.P. Cox, J. Jung, P. McDaniel, and A.N. Sheth, *TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones*, OSDI, 2010.
- M. Egele, C. Kruegel, E. Kirda, and G. Vigna, *Pios: Detecting privacy leaks in ios applications*, NDSS, 2011.
- Zhiqiang Lin, Xuxian Jiang, Dongyan Xu, and Xiangyu Zhang, *Automatic protocol format reverse engineering through context-aware monitored execution*, Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08) (San Diego, CA), February 2008.
- Long Lu, Zhichun Li, Zhenyu Wu, Wenke Lee, and Guofei Jiang, *Chex: statically vetting android apps for component hijacking vulnerabilities*, Proceedings of the 2012 ACM conference on Computer and communications security, ACM, 2012, pp. 229–240.
- Justin Ma, Kirill Levchenko, Christian Kreibich, Stefan Savage, and Geoffrey M. Voelker, *Unexpected means of protocol inference*, Proceedings of the 6th ACM SIGCOMM on Internet measurement (IMC'06) (Rio de Janeriro, Brazil), ACM Press, 2006, pp. 313–326.
- Nariman Mirzaei, Sam Malek, Corina S Păsăreanu, Naeem Esfahani, and Riyadh Mahmood, *Testing android apps through symbolic execution*, ACM SIGSOFT Software Engineering Notes **37** (2012), no. 6, 1–5.

References III

- *Ui/application exerciser monkey*, https://developer.android.com/tools/help/monkey.html.
- Aravind Machiry, Rohan Tahiliani, and Mayur Naik, *Dynodroid: An input generation system for android apps*, Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ACM, 2013, pp. 224–234.
- Paolo Milani Comparetti, Gilbert Wondracek, Christopher Kruegel, and Engin Kirda, *Prospex: Protocol Specification Extraction*, IEEE Symposium on Security & Privacy (Oakland, CA), 2009, pp. 110–125.
- James Newsome, David Brumley, Jason Franklin, and Dawn Song, *Replayer: Automatic protocol replay by binary analysis*, Proceedings of the 13th ACM Conference on Computer and and Communications Security (CCS'06), 2006.
- Vaibhav Rastogi, Yan Chen, and William Enck, *Appsplayground: Automatic security analysis of smartphone applications*, Proceedings of the Third ACM Conference on Data and Application Security and Privacy (New York, NY, USA), CODASPY '13, ACM, 2013, pp. 209–220.
- *A framework for analyzing and transforming java and android apps*, https://sable.github.io/soot/.
- David Sounthiraraj, Justin Sahs, Garrett Greenwood, Zhiqiang Lin, and Latifur Khan, *Smv-hunter: Large scale, automated detection of ssl/tls man-in-the-middle vulnerabilities in android apps*, Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS'14) (San Diego, CA), February 2014.
- Michelle Y Wong and David Lie, *Intellidroid: A targeted input generator for the dynamic analysis of android malware*, Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS'16) (San Diego, CA), February 2016.

References IV

Gilbert Wondracek, Paolo Milani, Christopher Kruegel, and Engin Kirda, *Automatic network protocol analysis*, Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08) (San Diego, CA), February 2008.

Chaoshun Zuo and Zhiqiang Lin, *Exposing server urls of mobile apps with selective symbolic execution*, Proceedings of the 26th World Wide Web Conference (WWW'17) (Perth, Australia), April 2017.

Chaoshun Zuo, Wubing Wang, Rui Wang, and Zhiqiang Lin, *Automatic forgery of cryptographically consistent messages to identify security vulnerabilities in mobile services*, Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS'16) (San Diego, CA), February 2016.

Yunhui Zheng, Xiangyu Zhang, and Vijay Ganesh, *Z3-str: A z3-based string solver for web application analysis*, Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ACM, 2013, pp. 114–124.