

RSA CONFERENCE
ASIA PACIFIC **2013**

STAMP: AN AUTOMATED UNKNOWN ZERO-DAY VULNERABILITY DISCOVERY SYSTEM FOR MOBILE PLATFORMS

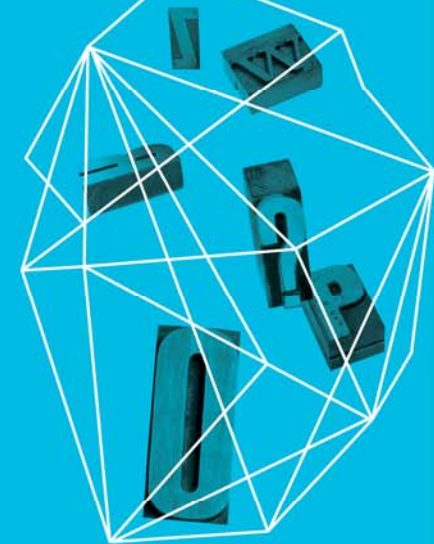
Dr. S. P. T. Krishnan

Institute for Infocomm Research

Ms. Seetha M. J.

Institute for Infocomm Research

Security in
knowledge



Session ID: MBS-W03

Session Classification: Intermediate

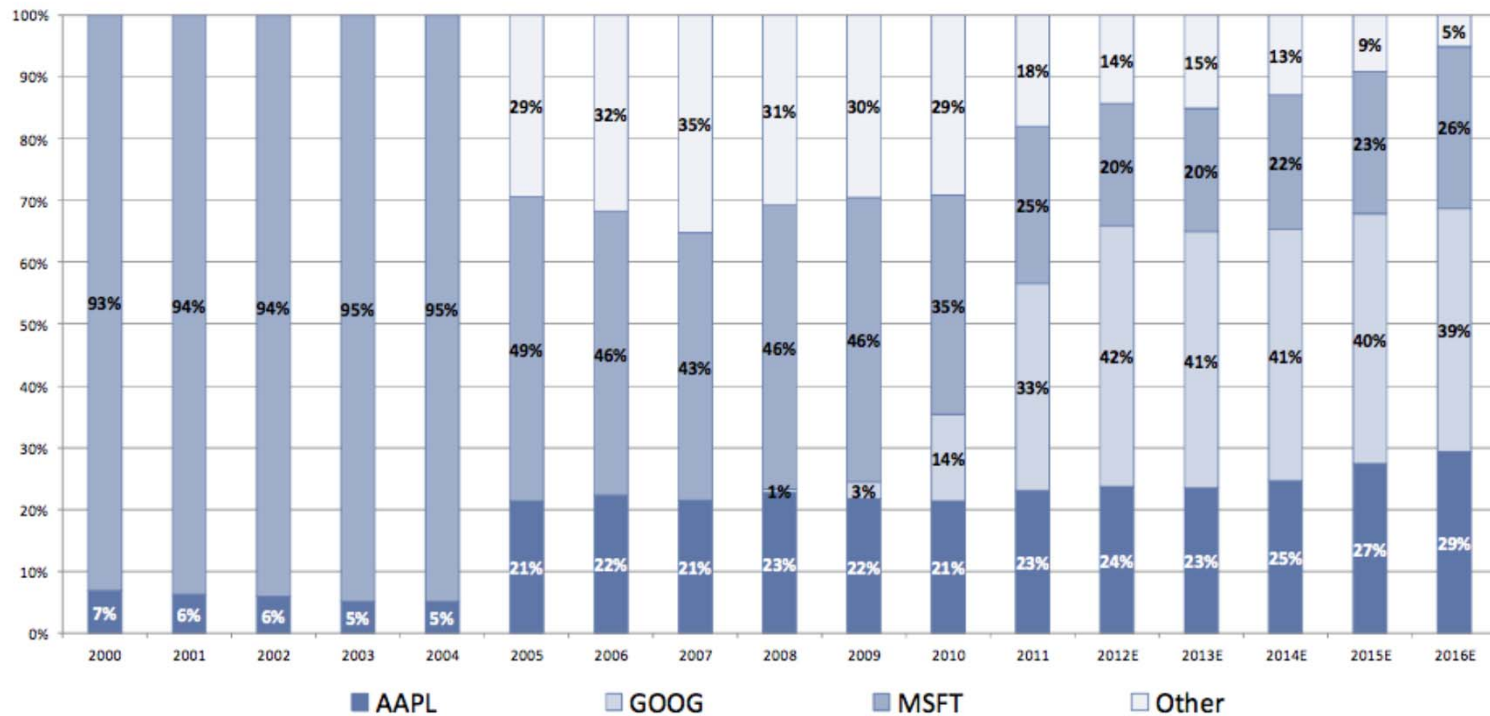
— Presentation Highlights

- ▶ The rise of Mobile OS
- ▶ The origin of security issues in mobile platforms
- ▶ STAMP
 - ▶ Server-side components
 - ▶ Client-side components
- ▶ Demo(s)

The Rise of Mobile OS

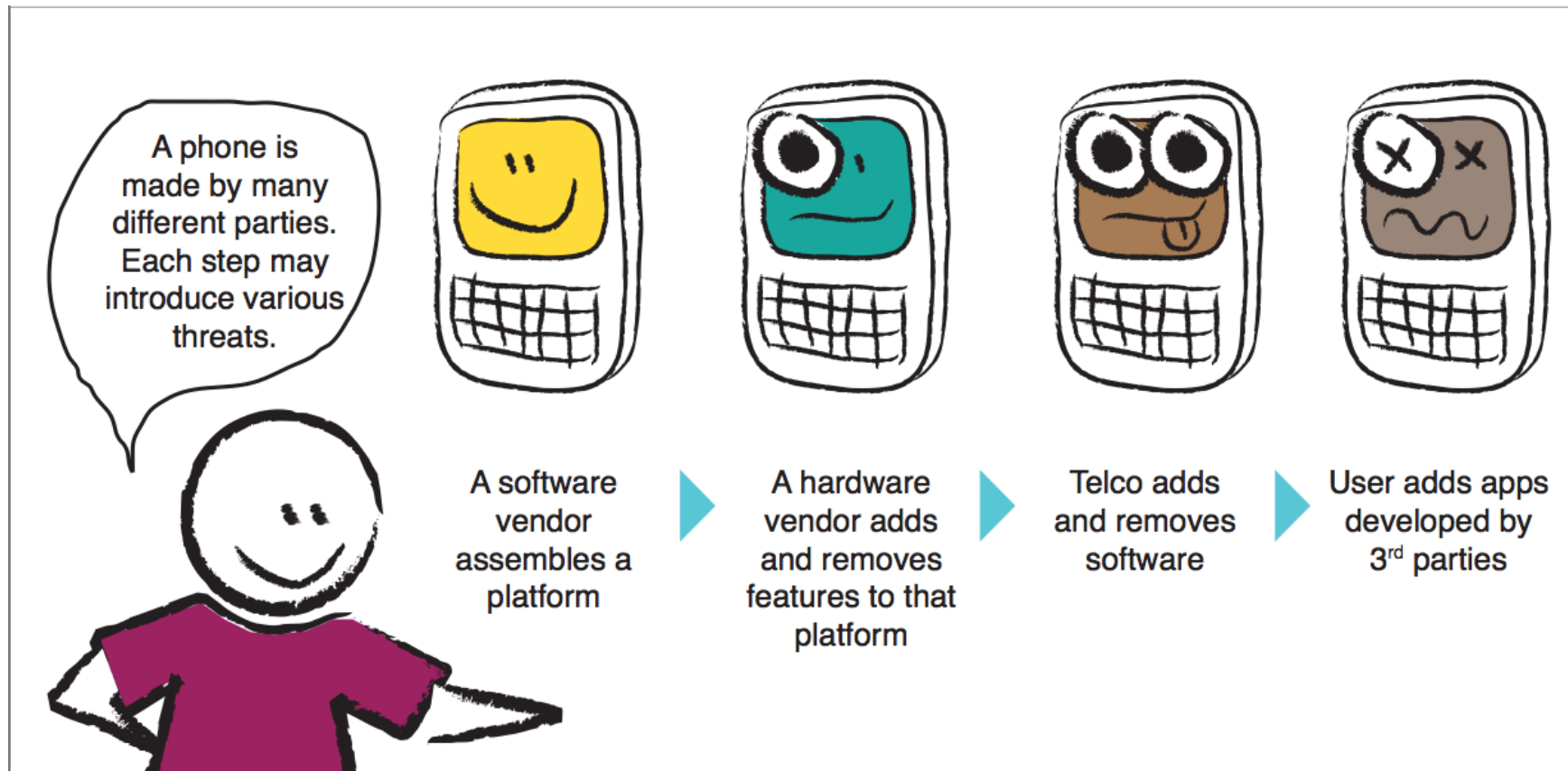
Exhibit 1: Vendor share of consumer compute, 2000-2016E

Shift from single-vendor dominance (MSFT) to multiple vendors (AAPL, GOOG, MSFT, Other)

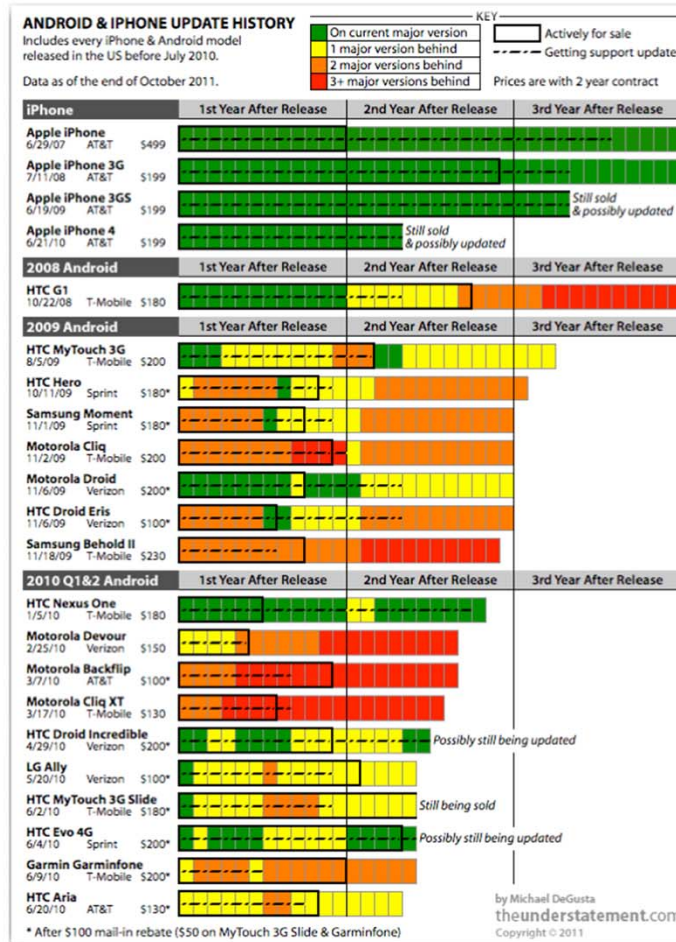


Source: IDC, Goldman Sachs Research.

— Mobile is Made Differently




Mobile OS Update History



— The Problem – 0 Day Vulnerability



The screenshot shows a web browser window with the address bar displaying www.zdnet.com/blog/security/google-android-vulnerable-to-drive-by-browser-.... The page title is "Google Android vulnerable to drive-by browser exploit". The author is Ryan Naraine, writing for Zero Day on October 27, 2008. The article summary states that the Google Android operating system is vulnerable to a serious security vulnerability that allows malicious hackers to launch drive-by browser attacks. The article also mentions that technical details of the vulnerability are being kept under wraps until a patch is available. The article includes social media sharing options for Comments (16), Votes (0), Like (1), Tweet (2), and Share. A small Android logo is visible next to the main text.

Topic: [Android](#) Follow via:  

Google Android vulnerable to drive-by browser exploit

Summary: *The Google Android operating system is vulnerable to a serious security vulnerability that allows malicious hackers to launch drive-by browser attacks, according to alert from a security research outfit. Technical details of the vulnerability, which occurs because Google Android uses an unpatched open-source software package, is being kept under wraps until a patch is available.*

By  Ryan Naraine for Zero Day | October 27, 2008 -- 10:38 GMT (18:38 SGT)
[Follow @ryanaraine](#)

Comments 16 Votes 0 Like 1 Tweet 2 Share more +

 The Google Android operating system is vulnerable to a serious security vulnerability that allows malicious hackers to launch drive-by browser attacks, according to alert from a security research outfit.

Technical details of the vulnerability, which occurs because Google Android uses an unpatched open-source software package, is being kept under wraps until a patch is available.

[**SEE: Android security team appeals to hackers**]

Google was notified of this issue on October 20th, 2008.

— Zero Day Vulnerability / Threat

- ▶ A zero-day vulnerability is a type of vulnerability that is discovered by an external entity (such as a hacker) and disclosed to the developer
- ▶ A zero-day attack exploits a previous unknown vulnerability i.e., an exploit code is made available before a patch is created by the developer

— What is Fuzzing?

- ▶ Fuzzing is a black-box robustness testing technique used to reveal unknown zero-day vulnerabilities by providing unexpected inputs.
- ▶ Three known types of fuzzing
 - ▶ Mutation-based fuzzing
 - ▶ Model-based fuzzing
 - ▶ Generational fuzzing

— AV Solutions are Ineffective



The screenshot shows a web browser window with the address bar displaying a URL. The page header features the logo for 'The Register' with the tagline 'Biting the hand that feeds IT'. The main content of the page is a news article titled 'Anti-virus products are rubbish, says Imperva' by Richard Chirgwin, dated 1st January 2013. The article discusses a study by Imperva that found anti-virus products are ineffective, with initial detection rates as low as 5% and many solutions taking up to a month to update their signatures.

Original URL:
http://www.theregister.co.uk/2013/01/01/anti_virus_is_rubbish/

Anti-virus products are rubbish, says Imperva
'Spend not proportional to effectiveness'
By [Richard Chirgwin](#)
Posted in [Security](#), 1st January 2013 21:15 GMT

[Do users have enough power? Take part in The Register's latest survey!](#)

A study released in December by US security outfit Imperva has tipped a bucket on the multi-billion-dollar anti-virus industry, claiming that initial detection rates are as low as five percent, and concluding that enterprise and consumer anti-virus spend "is not proportional to its effectiveness".

Working in conjunction with students from the Technion-Israel Institute of Technology, the company tested 82 malware samples against 40 anti-virus products including offerings from Microsoft, Symantec, McAfee and Kaspersky.

The test revealed that while catalogued viruses are well-detected, "less than 5% of anti-virus solutions in the study were able to initially detect previously non-cataloged viruses and that many solutions took up to a month or longer following the initial scan to update their signatures."

Interestingly, the study revealed that virus writers improve their chance of evading detection by keeping a low profile. If

Security Testing Arsenal for Mobile Platforms (STAMP)



— STAMP's Key Features

- ▶ Systematic process & system to discover zero-day vulnerabilities
- ▶ Complete tool chain to perform intelligent fuzzing of multiple remote targets
- ▶ Mundane & repetitive tasks are automated
- ▶ STAMP is our 3rd generation tool chain
 - ▶ 1st generation in 2010
- ▶ Performance Statistics
 - ▶ Tested with up to 300 simultaneous SUTs
 - ▶ ~50 million inputs tested
 - ▶ ~50K malicious inputs discovered

— STAMP Components

- ▶ *Server-side*

- ▶ **STAMP**: Security Testing Arsenal for Mobile Platforms
- ▶ **SFD**: Seed File Downloaders
- ▶ **SFAT**: Seed File Analysis Tool
- ▶ **FEET**: Fuzzing Engine Evaluation Tool
- ▶ **CACE**: Crash Analysis & Classification Engine

- ▶ *Client-side*

- ▶ **SOFT**: STAMP On-device Fuzzing Toolchain
- ▶ **SODA**: STAMP On-device Debugger & Agent

**RSA[®]CONFERENCE
ASIA PACIFIC 2013**

STAMP Server-side components



— STAMP Components – Server-side

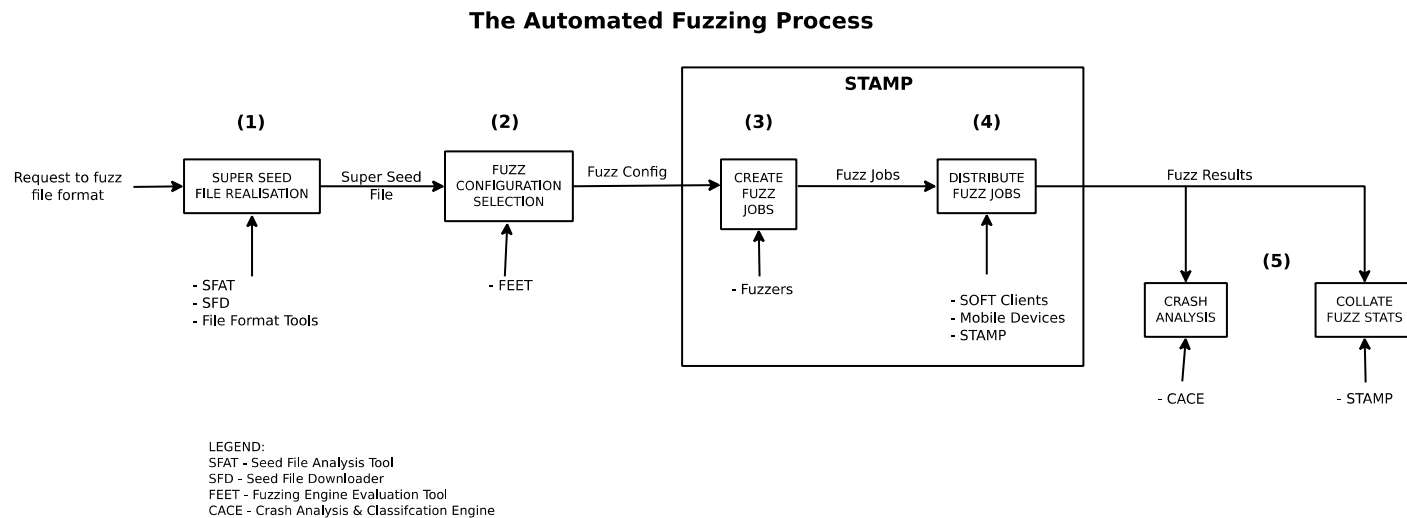
▶ **STAMP** Server

- ▶ Fat server design (centralized + intelligence)
- ▶ Fuzzing jobs generation & packaging (Fixed size – 10 MB)
- ▶ Fuzzing jobs assignment (default, on demand)
- ▶ Jobs optimized for mobile networks (key frames + diffs method)
- ▶ HTTPS (CA Signed) – Web console, Device access

▶ **CACE** (**C**rash **A**nalysis & **C**lassification **E**ngine)

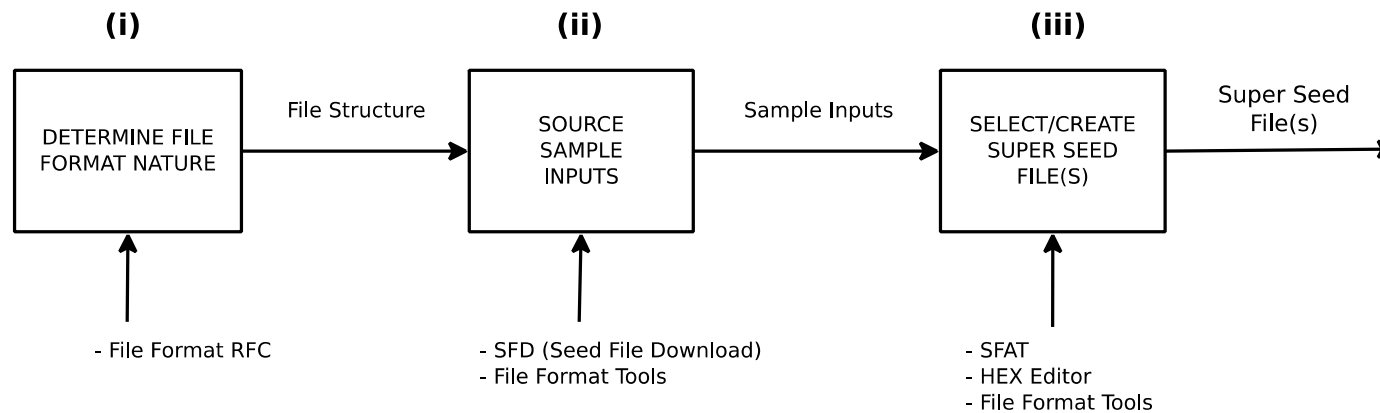
- ▶ A rule-based software to automatically analyze crash files to identify the source of crash
 - ▶ Location in binary (for iOS and Android)
 - ▶ Location in source (for Android)

The Exploitation Pipeline



— It all begins with a good seed

SUPER SEED FILE REALISATION (1)



LEGEND:

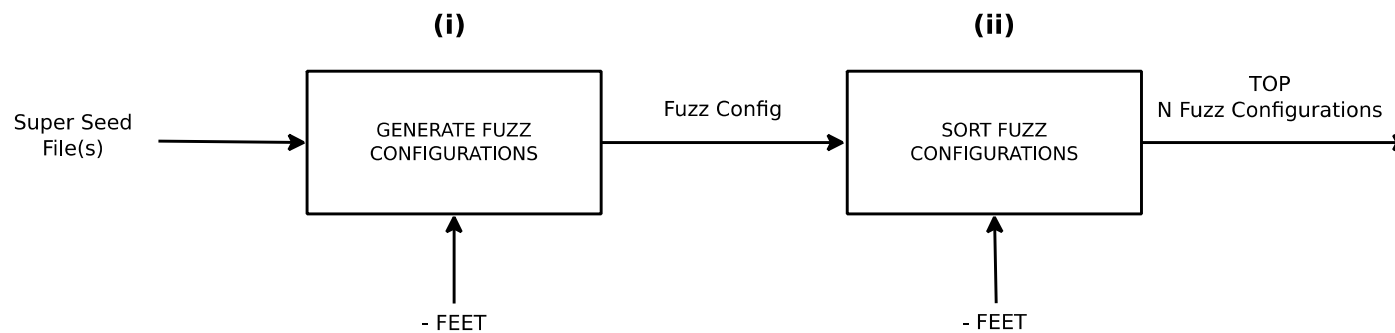
SFAT - Seed File Analysis Tool

SFD - Seed File Downloader

RFC - Request For Comments

— Which Fuzzing Engine to use ?

FUZZ CONFIGURATION SELECTION (2)



LEGEND:
FEET - Fuzzing Engine Evaluation Tool

— FEET – FE Configuration Selection

```
hgao@h: ~/Project/FE2_5_GetBestConfigurations_20130217
File Edit View Terminal Help
Operators being used to fuzz the files are : ['1', '2', '3', '4', '5', '6', '7',
'8', '9', '10']
===== FE2_5 JOB SCHEDULER STARTED =====
[FE2_5 SCHEDULER][FUZZING] Fuzzing job id 1
Begin fuzzing of flower-palette-04.tif...
Fuzzing completed
Checking for duplicates...
Checking completed...
Job ID 1, seed file: flower-palette-04.tif
-----
Total fuzzes: 10000
uniformity: 6.01740879347e-14
Percent uniqueness: 87.08
percentGlobalUniqueness: 74.16
numberMutations: 159897
[JOB SCHEDULER][EXIT] Exiting FE2_5 Job Scheduler
===== FE2_5 JOB SCHEDULER ENDED =====
Operators being used to fuzz the files are : ['1', '2', '3', '4', '5', '6', '7',
'8', '9', '10']
===== FE2_5 JOB SCHEDULER STARTED =====
[FE2_5 SCHEDULER][FUZZING] Fuzzing job id 1
Begin fuzzing of flower-palette-04.tif...
```

FEET – FE Best Configurations

mp3_wav2_sorted_33495_configs.csv - OpenOffice.org Calc

File Edit View Insert Format Tools Data Window Help

Liberation Sans 10

A1 $f(x)$ Σ = uniformity

	A	B	C	D	E	F	G	H	I
1	uniformity	random-seed	percentGlobalUniqueness	operator	percentUniqueness	baseline	denom	total-fuzz	changes
2	0.999999996097589	524287	100	1	100	mp3_wav2	50	10000	110555728206
3	0.9999999969587047	127	100	1	100	mp3_wav2	50	10000	110544003318
4	0.99999977042988	262147	100	1	100	mp3_wav2	50	10000	110497961036
5	0.999999731278138	16381	100	1	100	mp3_wav2	50	10000	110530910012
6	0.999998999206608	1000003	100	1	100	mp3_wav2	50	10000	110453451196
7	0.99999814495609	509	100	1	100	mp3_wav2	50	10000	110477012400
8	0.999996624951217	32771	100	1	100	mp3_wav2	50	10000	110558539773
9	0.99998026282119	2053	100	1	100	mp3_wav2	50	10000	110474662885
10	0.999933701259271	8191	100	1	100	mp3_wav2	50	10000	110457569759
11	0.999923809518961	4099	100	1	100	mp3_wav2	50	10000	110483829103
12	0.999783603685626	131071	100	1	100	mp3_wav2	50	10000	110564519682
13	0.999691684054731	31	100	1	100	mp3_wav2	50	10000	110509845934
14	0.999630530490245	65537	100	1	100	mp3_wav2	50	10000	110568481553
15	0.999174425355493	67	100	1	100	mp3_wav2	50	10000	110525109632
16	0.996299477693526	2053	100	4	100	mp3_wav2	100	10000	285125260
17	0.995654090898649	1021	100	1	100	mp3_wav2	50	10000	110484199250
18	0.995266938955892	257	100	1	100	mp3_wav2	50	10000	110603270964
19	0.985407842298762	67	100	5	100	mp3_wav2	5	10000	6002226640
20	0.985407842298762	67	100	6	100	mp3_wav2	5	10000	6002200670
21	0.976217710912799	524287	100	6,9	100	mp3_wav2	100	10000	99941299
22	0.975065729580689	524287	100	1	100	mp3_wav2	10	10000	22108223152
23	0.96952146492743	2053	100	5	100	mp3_wav2	50	10000	619433807

— Fuzzing Engine Evaluation Tool

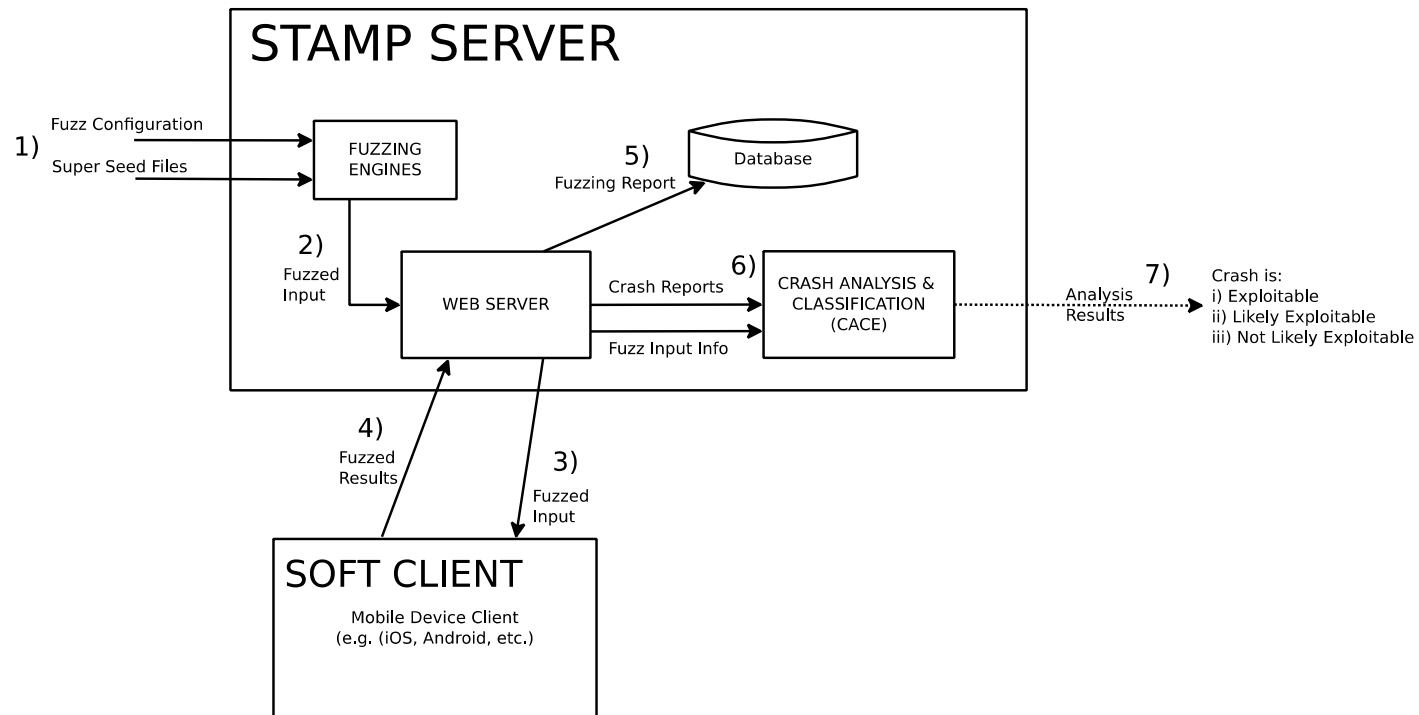
▶ Uniformity

- ▶ Mutation locations should be uniformly distributed
- ▶ Uniformity: the "goodness to fit" between the observed mutation locations and expected mutation locations
- ▶ Chi-Square Test to measure the uniformity
- ▶ Uniformity is important to ensure that different locations will be mutated by the fuzzing engines

▶ Uniqueness

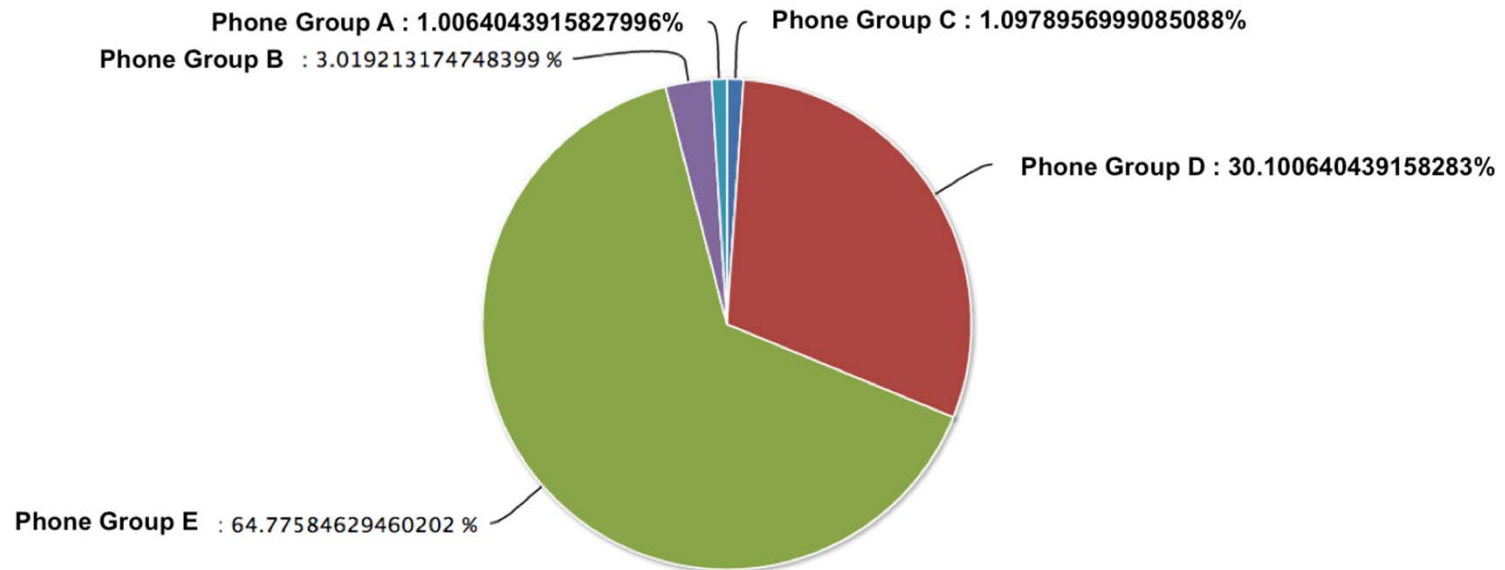
- ▶ Local uniqueness – within one fuzzing job
- ▶ Global uniqueness – within one fuzzing engine; across jobs
- ▶ Universal uniqueness – across all fuzzing engines

The STAMP Server Architecture



— STAMP – Device Distribution

Percentage of verified crashes across all phone groups



STAMP – Crash Statistics



— STAMP – Detailed Statistics

Crash occurrence across all phone models

Search Search

• by image name

File Name:	Model A	Model B	Job ID	REMARKS
jpg_test1-38.jpg			<u>2</u>	N.A
jpg_test1-242.jpg			<u>2</u>	N.A
jpg_test1-390.jpg			<u>2</u>	N.A
jpg_test1-1008.jpg			<u>2</u>	N.A
jpg_test1-1224.jpg			<u>2</u>	N.A

Page 1 of 15 > >>

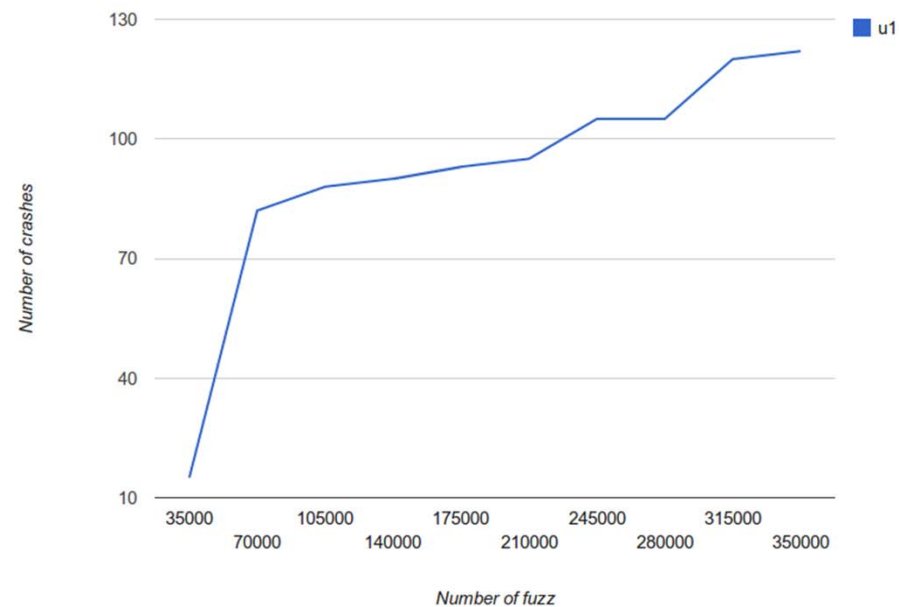
— STAMP – Fuzzing Gradient

Home > Fuzzing Gradient > u1

Select a phone model

Phone model:

Fuzzing gradient of phone model: u1



**RSA[®]CONFERENCE
ASIA PACIFIC 2013**

STAMP Client-side components



— STAMP Components – Client-side

▶ **SOFT**

- ▶ Thin-client design (Easy of porting to new platforms)
- ▶ Multi-threaded architecture (download, upload, fuzzing)
- ▶ Supported platforms: Android, iOS, Symbian
 - ▶ Beta: Blackberry, BB10, Windows phone 7

▶ **SODA**

- ▶ Fully contained (non/interactive) debugger on-device
- ▶ Abstraction on top of mobile platform's debugging capabilities
- ▶ Symbian: Kernel-module (OEM signed)
- ▶ Android / iOS: Interface to GDB

**RSAC[®]CONFERENCE
ASIA PACIFIC 2013**

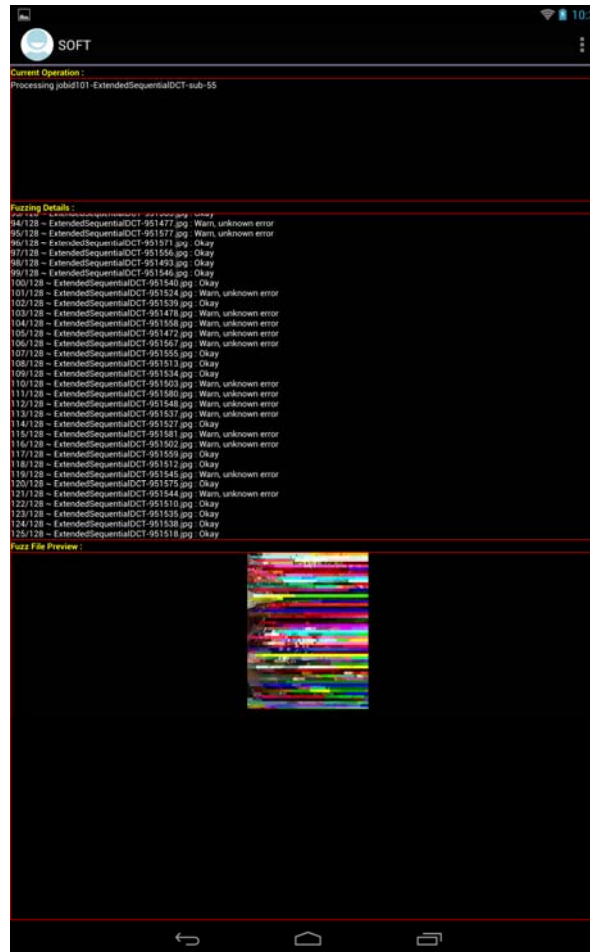
STAMP On-device Fuzzing Toolchain (SOFT)



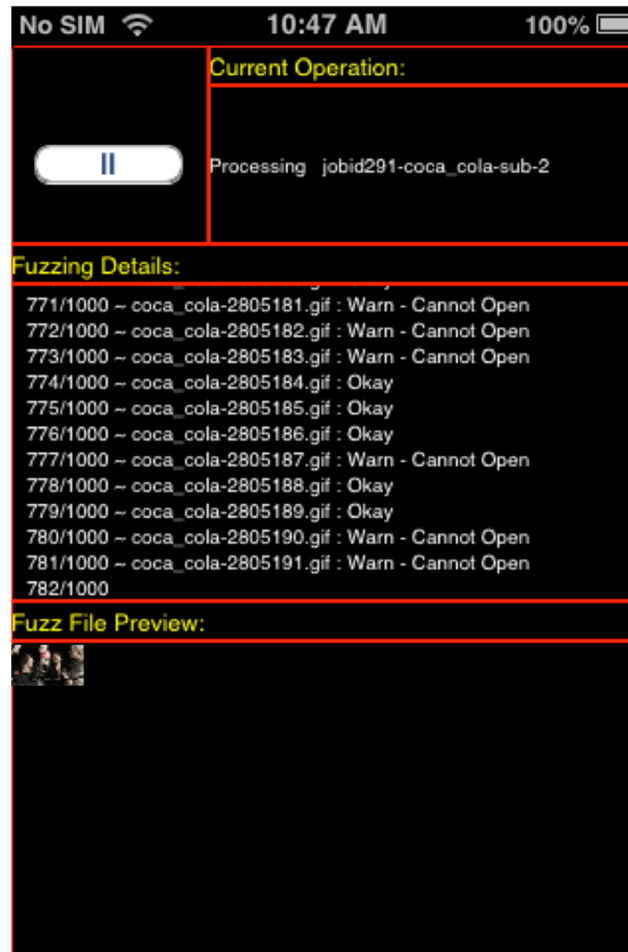
— SOFT on Android Phone



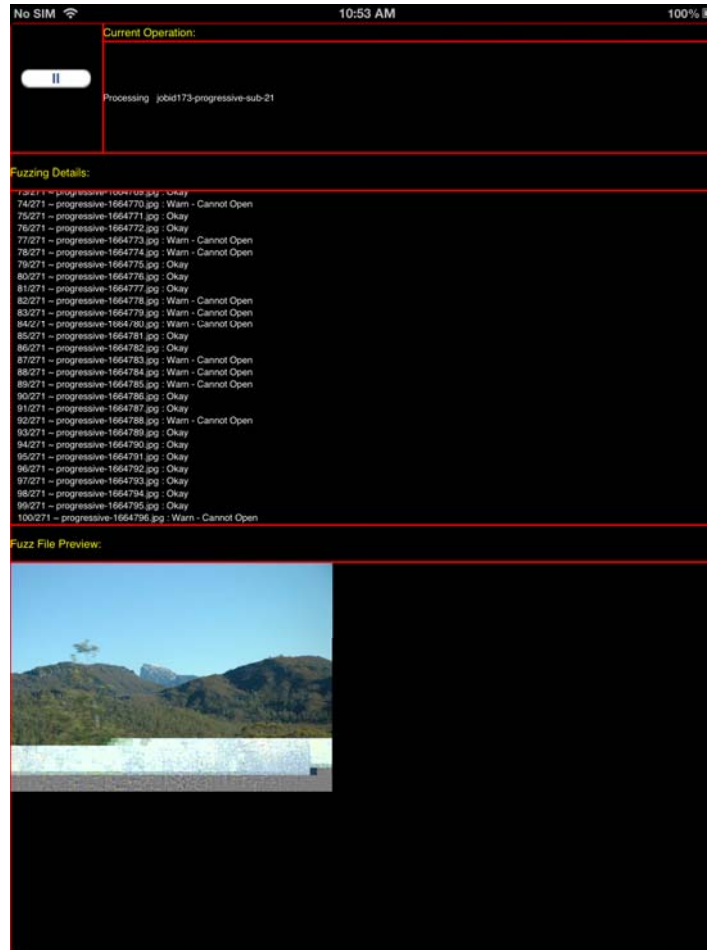
SOFT on Android Tablet



— SOFT on iPhone



— SOFT on iPad



STAMP On-device Debugger & Agent (SODA)



— SODA: Multi-Platform Support

STAMP On-device Debugger & Agent



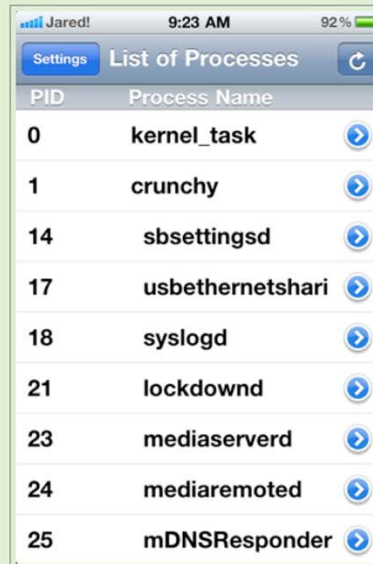
— SODA: On-device Task Manager

STAMP On-device Debugger & Agent



SODA

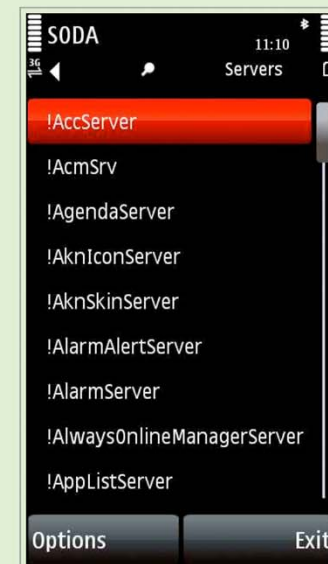
PID	Process Name
96	system
160	com.android.systemui
168	com.google.android.inputmethod.latin
177	com.android.phone
184	com.android.launcher
207	com.google.process.gapps
226	android.process.acore
264	com.android.settings
288	com.android.vending
331	com.lookout
371	android.process.media
447	org.onaips.vnc



Jared! 9:23 AM 92%

Settings List of Processes

PID	Process Name
0	kernel_task
1	crunchy
14	sbsettingsd
17	usbethernetshari
18	syslogd
21	lockdownd
23	mediaserverd
24	mediaremoted
25	mDNSResponder



SODA 11:10

Servers

!AccServer
!AcmSrv
!AgendaServer
!AknIconServer
!AknSkinServer
!AlarmAlertServer
!AlarmServer
!AlwaysOnlineManagerServer
!AppListServer

Options Exit

— SODA: On-device Debugger

STAMP On-device Debugger & Agent



— SODA: Remote Debugging Agent

STAMP On-device Debugger & Agent

The screenshot displays the SODA remote debugging interface. On the left, a window titled 'SODA' shows a 'Select a command from the following:' dropdown menu with options: Suspend, Resume, Terminate, and Refresh. The 'Refresh' option is currently selected. To the right of this menu is a 'Thread list' containing a scrollable list of process names and their PIDs, such as 'bsengine.exe[2000f83e]0001:bsengine' and 'Calculator[10005902]0001:Calculator'. The 'Refresh' option is highlighted in blue. On the right side of the interface, a 'SODA client' window is visible, showing the text 'Select an option from the 'Options' menu', 'Connecting to server..', and 'Connected to server'. Below this text are two buttons: 'Options' and 'Exit'.

SODA Dumps

RETC Administration Interface Welcome, **Seetha**. [Update password](#) / [Update profile](#) / [Log out](#)

[Home](#) > [View dumps](#)

Crash image information: jpg_test1-390.jpg tested on RM-356 (40.0.005)

Sodadumps for IMEI: 354182027933884 [+Download dumps](#)

```
Image Name: jobid2-jpg_test1-sub-3\jpg_test1-390.jpg
Date: 18-8-2010
Time: 16:38:46:644625
Thread_Name: SOFT[ef8a2577]0001::SOFT
Process_ID: 723
Exception_Number: 44
Panic_Category: USER
Event_Type: 2
Process_Priority: 2
UID1: 0x[1000007A]
UID2: 0x[100039CE]
UID3: 0x[EF8A2577]
CodeBase: 0x[79094000] (3220)
CodeBaseEndAddress: 0x[79094C94]
ConstDataBase: 0x[0] (0)
InitialisedDataBase: 0x[400000] (8)
UninitialisedDataBase: 0x[400008] (0)
DLL Count: 36
Thread_ID: 724
ThreadPriority: -5
ThreadType: 3
RunAddress: 0x[79094000] (4096)
```

**RSA[®]CONFERENCE
ASIA PACIFIC 2013**

**STAMP DEMO(S)
and
Q & A**



— STAMP Demo(s)

- ▶ SOFT on iOS and Android
 - ▶ <http://youtu.be/lkv-6JXvO4o>
 - ▶ iOS unknown vulnerability discovered through STAMP
 - ▶ <http://youtu.be/rRD0h5tDlu0>
- (The above URLs are only available until the talk)*
- ▶ STAMP Web Console Live Walkthrough
 - ▶ STAMP Auto-installer

— A NOTE TO ORGANISER

- ▶ **STAMP demo** will be conducted **Live** using the following equipment.
 1. Mac Book Pro + power adapter (with VGA Adapter)
 1. *For security reasons, we prefer to use our laptop when logging to our live portal.*
 2. Visualizer + power adapter (with VGA output)
 1. *For showing the SOFT client on mobile phones. We can bring this if unavailable.*
 3. 2x Mobile phones (with power chargers)
 1. *We will bring the phones.*
 4. *A 4-socket power extension box is required.*