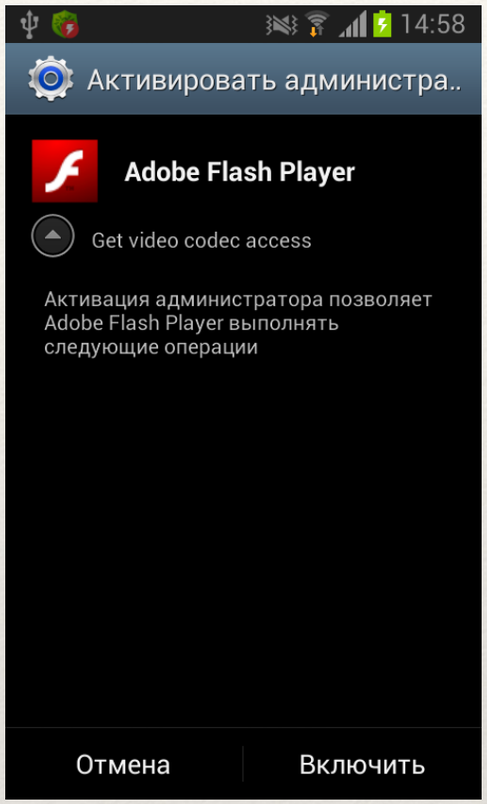
USENIX Security, 2014

a bayesian approach to privacy enforcement in smartphones

Omer Tripp
IBM Research, NY

Julia Rubin
IBM Research, Haifa

motivation



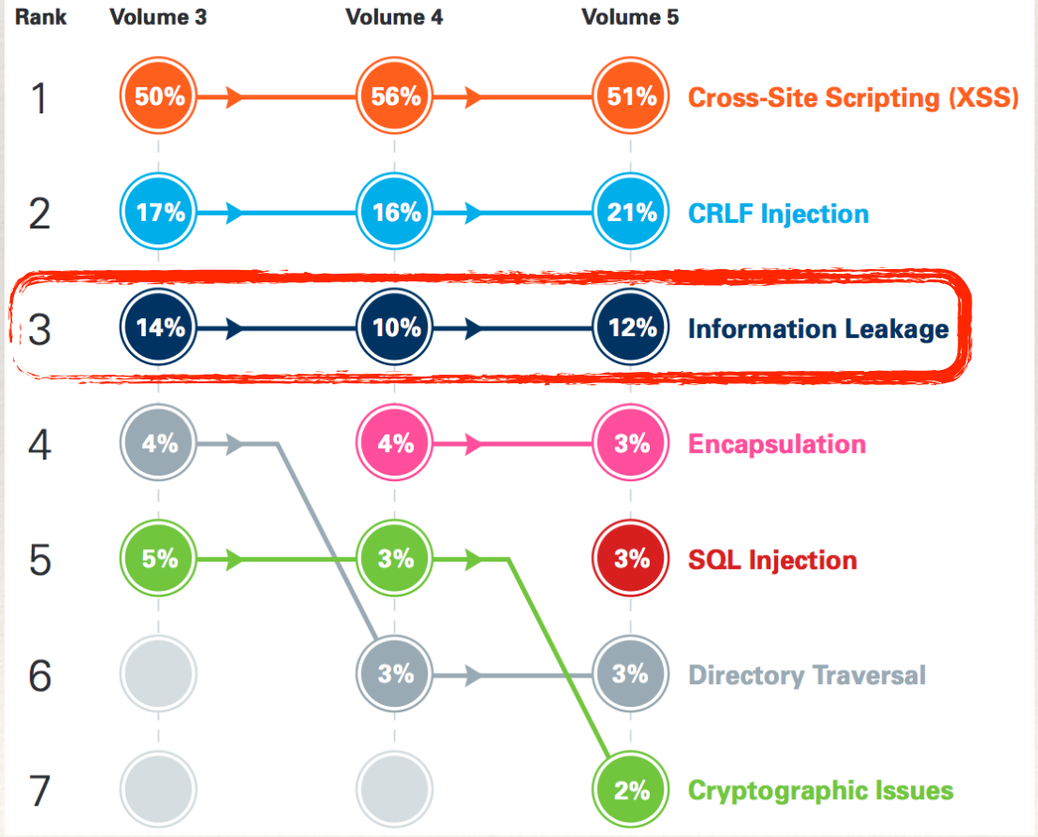
* *http://hackread.com/android-malware-steals-credit-card-information/*

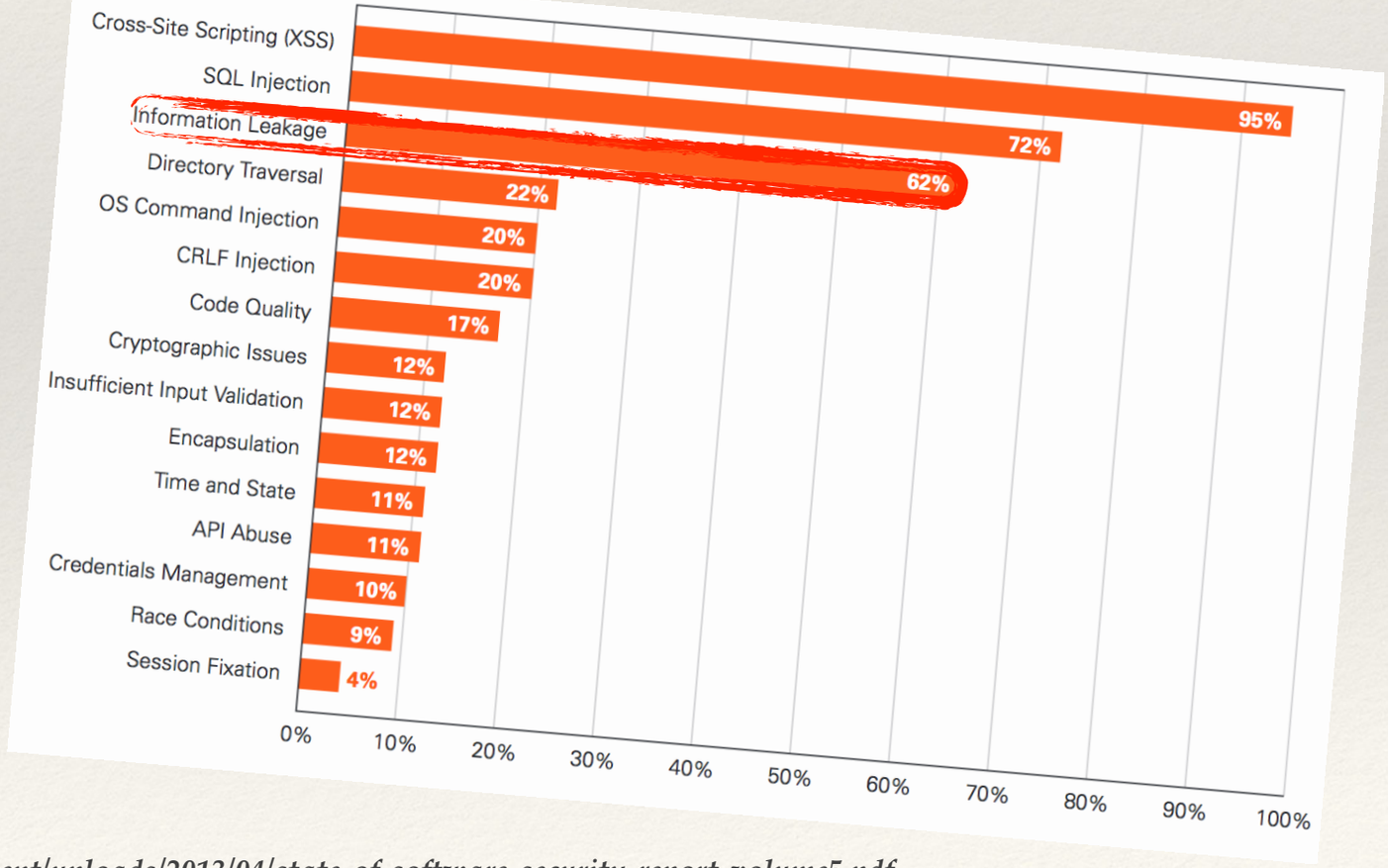


* *http://www.theguardian.com/world/2014/jan/27/nsa-gchq-smartphone-app-angry-birds-personal-data*

motivation

Android		iOS		Java ME	
CRLF Injection	37%	Information Leakage	62%	Cryptographic Issues	47%
Cryptographic Issues	33%	Error Handling	20%	Information Leakage	47%
Information Leakage	10%	Cryptographic Issues	7%	Directory Traversal	3%
SQL Injection	9%	Directory Traversal	6%	Insufficient Input Validation	2%
Time and State	4%	Buffer Management Errors	3%	Credentials Management	<1%





* http://blackdiamondsolutions.com/wp-content/uploads/2013/04/state-of-software-security-report-volume5.pdf

illustrative example

```
Address address = getFromLocation(getLocation()); // source

String country = address.getCountry();

String mImsi = …; // source

// 6 digits <= IMSI (MCC+MNC+MSIN) <= 15 (usually 15)

if (mImsi != null && (mImsi.length()<6 || mImsi.length()>15)) {

      loge(“invalid IMSI “ + mImsi); mImsi = null; } // sink

log(   “IMSI: “ + mImsi.substring(0,6) +

       “xxxxxxxxx” + “ , country: “ + country); // sink
```

(slightly revised from: `com.android.internal.telephony.cdma.RuniRecords`*)*

illustrative example

```
mImsi = "4046855056012340"

loge("invalid IMSI "4046855056012340")
```

versus

```
mImsi = "4046855056012340"

address = { "1101 Kitchawan Rd", "Yorktown", "US", … }

log("IMSI "404685xxxxxxxxx , country = US")
```

illustrative example

```
mImsi = "4046855056012340"

address = { "1101 Kitchawan Rd", "Yorktown", "US", … }


log("IMSI 404685xxxxxxxxx , country = US")
```

versus

```
mImsi = "4046855056012340"

address = { "1101 Kitchawan Rd", "Yorktown", "US", … }


log("IMSI 404685xxxxxxxxx , country = US")
```

illustrative example

```
mImsi = "4046855056012340"

loge("invalid IMSI "4046855056012340")
```

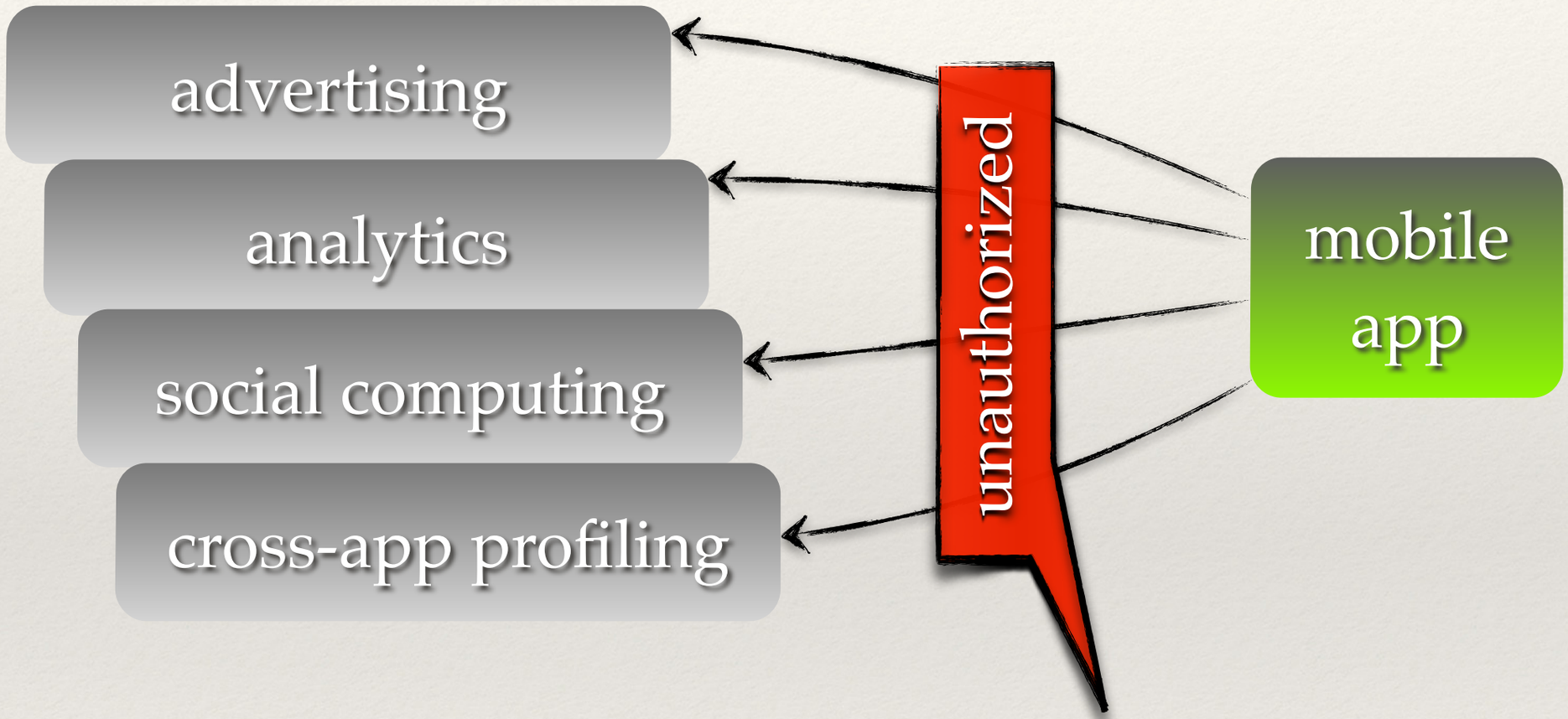
runtime enforcement

```
mImsi = "4046855056012340"

address = { "1101 Kitchawan Rd", "Yorktown", "US", … }

log("IMSI "404685xxxxxxxxxx , country = US")
```

threat model



standard transformations (if any): Base64, MD5, plain, ….

a bayesian perspective

$$\Pr(X \mid Y) = \frac{\Pr(Y \mid X)\Pr(X)}{\Pr(Y)}$$



Rev. T. Bayes

a bayesian perspective

Pr(`log(s)` is illegitimate | [X=x , Y=y , Z=z , …])

versus

Pr(`log(s)` is legitimate | [X=x , Y=y , Z=z , …])

where

```
s="IMSI 404685xxxxxxxxxx , country = US"
```

bayes / taint

$\Pr^{\#}(\texttt{log(s)}$ is illegitimate $\mid$ $\texttt{s}$ is tainted$) = 1$

versus

$\Pr^{\#}(\texttt{log(s)}$ is legitimate $\mid$ $\texttt{s}$ is tainted$) = 0$

where

```
s="IMSI 404685xxxxxxxxxx , country = US"
```

bayes / taint

$\Pr^{\#}(\texttt{log(s)}$ is illegitimate $\mid$ $\texttt{s}$ is tainted$) = \mathbf{1}$

versus

$\Pr^{\#}(\texttt{log(s)}$ is legitimate $\mid$ $\texttt{s}$ is tainted$) = \mathbf{0}$

where

```
s="IMSI 404685xxxxxxxxxx , country = US"
```

too conservative

bayes / value-based features

$\Pr^{\#}(\texttt{log(s)}$ is illegitimate $\mid IMSI_{\texttt{s}}=r_1, Addr_{\texttt{s}}=r_2)$

versus

$\Pr^{\#}(\texttt{log(s)}$ is legitimate $\mid IMSI_{\texttt{s}}=r_1, Addr_{\texttt{s}}=r_2)$

where

```
s = "invalid IMSI 404685xxxxxxxxxx , country = US"
```

$Addr_{\texttt{s}} = distance(\texttt{s}, \{\texttt{"1101 Kitchawan Rd","Yorktown","US", …}\})$

$IMSI_{\texttt{s}} = distance(\texttt{s}, \texttt{"404685056012340"})$

bayes / value-based features

$\Pr^{\#}(\texttt{log(s)}$ is illegitimate $\mid IMSI_{\texttt{s}}=r_1, Addr_{\texttt{s}}=r_2)$

versus

$\Pr^{\#}(\texttt{log(s)}$ is legitimate $\mid IMSI_{\texttt{s}}=r_1, Addr_{\texttt{s}}=r_2)$

where

```
s = "invalid IMSI 404685xxxxxxxxxx , country = US"
```

$Addr_{\texttt{s}} =$ *distance*`(s, {"1101 Kitchawan Rd","Yorktown","US", …})`

$IMSI_{\texttt{s}} =$ *distance*`(s, "404685056012340")`

bayes / value-based features

$\Pr^{\#}(\texttt{log(s)}$ is illegitimate $\mid IMSI_{\texttt{s}}=r_1, Addr_{\texttt{s}}=r_2)$

versus

$\Pr^{\#}(\texttt{log(s)}$ is legitimate $\mid IMSI_{\texttt{s}}=r_1, Addr_{\texttt{s}}=r_2)$

where

```
s = "invalid IMSI 404685xxxxxxxxxx , country = US"
```

$Addr_{\texttt{s}} = distance(\texttt{s}, \{\texttt{"1101 Kitchawan Rd","Yorktown","US", …}\})$

$IMSI_{\texttt{s}} = distance(\texttt{s}, \texttt{"404685056012340"})$

bayes / value-based features

$\Pr^{\#}(\texttt{log(s)}$ is illegitimate $\mid IMSI_{\texttt{s}}=r_1, Addr_{\texttt{s}}=r_2)$

s *is tainted*

versus

$\Pr^{\#}(\texttt{log(s)}$ is legitimate $\mid IMSI_{\texttt{s}}=r_1, Addr_{\texttt{s}}=r_2)$

s *is tainted*

where

$\texttt{s} =$ `"invalid IMSI 404685xxxxxxxxxx , country = US"`

$Addr_{\texttt{s}} = distance(\texttt{s}, \{$`"1101 Kitchawan Rd","Yorktown","US", …`$\})$

$IMSI_{\texttt{s}} = distance(\texttt{s},$ `"404685056012340"`$)$

bayes / value-based features

$\Pr^{\#}(\texttt{log(s)}\text{ is illegitimate} \mid \mathit{IMSI}_{\texttt{s}}{=}r_1, \mathit{Addr}_{\texttt{s}}{=}r_2)$

versus

$\Pr^{\#}(\texttt{log(s)}\text{ is legitimate} \mid \mathit{IMSI}_{\texttt{s}}{=}r_1, \mathit{Addr}_{\texttt{s}}{=}r_2)$

where

$\texttt{s} = \text{“}\texttt{invalid IMSI } \mathit{404685}\texttt{xxxxxxxxxx , country = } \mathit{US}\text{”}$

$\mathit{Addr}_{\texttt{s}} = \mathit{distance}(\texttt{s}, \{\text{“}\mathit{1101\ Kitchawan\ Rd}\text{”},\text{“}\mathit{Yorktown}\text{”},\text{“}\mathit{US}\text{”}, \ldots\})$

$\mathit{IMSI}_{\texttt{s}} = \mathit{distance}(\texttt{s}, \text{“}\mathit{404685056012340}\text{”})$

value-based features — why?

- precision: capture inherent fuzziness of privacy domain
  - e.g.: *404685*`xxxxxxxxxx` vs *4046855056*`xxxxx` vs *404685505610234*
- robustness: can we do without taint tracking?
  - e.g.: $\Pr^{\#}(\ldots \mid IMSI_{\mathtt{s}}{=}r_1, Addr_{\mathtt{s}}{=}r_2)$ vs $\Pr^{\#}(\ldots \mid IMSI_{\mathtt{s}}{=}r_1, Addr_{\mathtt{s}}{=}r_2, \mathtt{s}$ is tainted)

value-based features – how?

value-based features – how?

- <u>assumption I:</u> distance metric **bounded** by arguments

$$d(x,y) <= \max(x,y)$$

value-based features – how?

- assumption I: distance metric **bounded** by arguments

$$d(x,y) <= \max(x,y)$$

- assumption II: feature values **bounded** from above

for all features f in F. exists **c**. $|[f]| < \mathbf{c}$

value-based features – how?

- assumption I: distance metric **bounded** by arguments

$$d(x,y) <= \max(x,y)$$

- assumption II: feature values **bounded** from above

for all features f in F. exists **c**. $|[f]| < \mathbf{c}$

- assumption III: comparisons **bounded** by feature value

in all comparisons $d(f,y)$. $|y| <= |[f]|$

value-based features – how?

$$\mathrm{ham}_{a,b}(|\boldsymbol{a}|,|\boldsymbol{b}|) = \#\mathrm{D}\{\, 0 <= \mathrm{i} <= |\boldsymbol{a}| : \boldsymbol{a}_i \mathrel{!}= \boldsymbol{b}_j \,\}$$

Hamming distance: min # of substitutions to transform ***a*** into ***b***

value-based features – how?

$$\mathrm{lev}_{a,b}(|a|,|b|) = \begin{cases} \max(i,j) & \text{if } \min(i,j)=0 \\ \min\begin{pmatrix} \mathrm{lev}(i-1,j)+1 \\ \mathrm{lev}(i,j-1)+1 \\ \mathrm{lev}(i-1,j-1) + [a_i \mathrel{!}= b_j] \end{pmatrix} & o/w \end{cases}$$

Levenshtein distance: min # of char edits to transform ***a*** into ***b***

value-based features – when?

$$\Pr^{\#}(\texttt{log(s)}\ \text{leg/illeg} \mid IMSI_{\texttt{s}}=r_1, Addr_{\texttt{s}}=r_2) =$$

$$\frac{\Pr^{\#}(IMSI_{\texttt{s}}=r_1, Addr_{\texttt{s}}=r_2 \mid \texttt{log(s)}\ \text{leg/illeg})\ \Pr^{\#}(\texttt{log(s)}\ \ \text{leg/illeg})}{\Pr^{\#}(IMSI_{\texttt{s}}=r_1, Addr_{\texttt{s}}=r_2)}$$

value-based features – when?

$$\Pr^{\#}(\texttt{log(s)}\ \text{leg/illeg} \mid \mathit{IMSI}_{\texttt{s}}=\mathrm{r}_1, \mathit{Addr}_{\texttt{s}}=\mathrm{r}_2) =$$

$$\frac{\Pr^{\#}(\mathit{IMSI}_{\texttt{s}}=\mathrm{r}_1, \mathit{Addr}_{\texttt{s}}=\mathrm{r}_2 \mid \texttt{log(s)}\ \text{leg/illeg})\ \Pr^{\#}(\texttt{log(s)}\ \text{leg/illeg})}{\Pr^{\#}(\mathit{IMSI}_{\texttt{s}}=\mathrm{r}_1, \mathit{Addr}_{\texttt{s}}=\mathrm{r}_2)}$$

never: same denominator in both leg and illeg expression

value-based features – when?

$$\Pr^{\#}(\texttt{log(s)}\ \text{leg/illeg} \mid \mathit{IMSI}_{\texttt{s}}{=}r_1, \mathit{Addr}_{\texttt{s}}{=}r_2) =$$

$$\frac{\Pr^{\#}(\mathit{IMSI}_{\texttt{s}}{=}r_1, \mathit{Addr}_{\texttt{s}}{=}r_2 \mid \texttt{log(s)}\ \text{leg/illeg})\ \Pr^{\#}(\texttt{log(s)}\ \text{leg/illeg})}{\cancel{\Pr^{\#}(\mathit{IMSI}_{\texttt{s}}{=}r_1, \mathit{Addr}_{\texttt{s}}{=}r_2)}}$$

offline: based on existing studies (e.g.: Hornyack et al. & Enck et al.)

value-based features – when?

$$\Pr^{\#}(\texttt{log(s) leg/illeg} \mid IMSI_{\texttt{s}}=r_1, Addr_{\texttt{s}}=r_2) =$$

$$\frac{\Pr^{\#}(IMSI_{\texttt{s}}=r_1, Addr_{\texttt{s}}=r_2 \mid \texttt{log(s) leg/illeg})\ \Pr^{\#}(\texttt{log(s) leg/illeg})}{\cancel{\Pr^{\#}(IMSI_{\texttt{s}}=r_1, Addr_{\texttt{s}}=r_2)}}$$

offline: manual analysis of 35 apps ($\#D\{X_i=x_{ij}\ \&\ Y=y_k\}\ /\ \#D\{Y=y_k\}$)

the BAYESDROID algorithm

onSrcStmt `r=src [p]`:

f := **getFeature** `src`

map source API to feature (e.g.: `getDeviceId()` > *IMEI)*

attach tag f to `r`

taint seed

onNormalStmt `r=nrm [p]`:

propagate feature tags according to data flow

taint propagation

onSinkStmt `r=snk [p]`:

$F=\{f > [\mathtt{p}_f]\}$:= **extractTags** `[p]` ; $v := \{\,\}$

tag-based feature/values extraction

for all $f > [\mathtt{p}_f]$ *in F.* { u := ref f ; $v[f] := d^*(u,[\mathtt{p}_f])$ }

similarity checking

alarm if $\Pr^{\#}(\mathtt{r\ snk\ [p]}\ \text{illeg} \mid v) > \Pr^{\#}(\mathtt{r\ snk\ [p]}\ \text{leg} \mid v)$

bayesian judgment

BayesDroid: beyond plain text

onSrcStmt `r=src [p]`:

f := **getFeature** `src`

attach tag f to `r`

onNormalStmt `r=nrm [p]`:

propagate feature tags according to data flow

onSinkStmt `r=snk [p]`:

$F=\{f > [\mathtt{p}_f]\}$:= **extractTags** `[p]` ; $v := \{\,\}$

for all $f > [\mathtt{p}_f]$ *in* F, ***t in T.*** { u := ***t*** (ref *f*) ; ***v[f]*** := $d^*(u,[\mathtt{p}_f])$ }

T: a set of std. transformations

alarm if $\Pr^{\#}(\mathtt{r\ snk\ [p]}\ \text{illeg} \mid v) > \Pr^{\#}(\mathtt{r\ snk\ [p]}\ \text{leg} \mid v)$

BAYESDROID: unbounded values

onSrcStmt `r=src [p]`:

f := **getFeature** `src`

attach tag f to `r`

onNormalStmt `r=nrm [p]`:

propagate feature tags according to data flow

onSinkStmt `r=snk [p]`:

$F=\{f > [\mathtt{p}_f]\}$:= **extractTags** `[p]` ; $v := \{\,\}$

for all $f > [\mathtt{p}_f]$ *in F, t in T.* { $\boldsymbol{u} := t$ (ref f) { $v[f] := \min\{\, d^*(\boldsymbol{u}, [\mathtt{p}_f], \mathbf{c}_f \,\})$ } }

$\mathbf{c}_f$: per-feat bound

alarm if $\Pr^{\#}(\mathtt{r\ snk\ [p]}\ \text{illeg} \mid v) > \Pr^{\#}(\mathtt{r\ snk\ [p]}\ \text{leg} \mid v)$

BayesDroid: taint-free tracking

onSrcStmt `r=src [p]`:

f := **getFeature** `src`

~~*attach tag f to* `r`~~

~~**onNormalStmt** `r=nrm [p]`:~~

~~*propagate feature tags according to data flow*~~

onSinkStmt `r=snk [p]`:

F={f > [$\mathtt{p}_f$]} := **mapAllValsUpToDepthK (**`[p]`,k**)** v := { } comp. w all vals of depth <= k

for all f > [$\mathtt{p}_f$] *in* F, t *in* T. { $\boldsymbol{u}$:= t (ref f) { $\boldsymbol{v[f]}$:= min { $d^*(\boldsymbol{u},[\mathtt{p}_f], \mathbf{c}_f$ }) }

alarm if $\Pr^{\#}$(`r snk [p]` illeg | v) > $\Pr^{\#}$(`r snk [p]` leg | v)

H1: accuracy

bayesian reasoning about data leakage, accounting explicitly for value similarity, is significantly more accurate than taint tracking

H1: methodology

- DroidBench: independent and publicly available privacy suite consisting of 50 test cases
- excluded 8 benchmarks that crash at startup and 5 that leak data via unreachable callbacks (e.g.: `onLowMemory()`)
- compared with the TaintDroid enforcement system

H1: results

Benchmark	BayesDroid				TaintDroid			
	TPs	FPs	FNs	acc	TPs	FPs	FNs	acc
ActivityCommunication1	1	0	0	1.0	1	0	0	1.0
ActivityLifecycle1								
AnonymousClass1	0	0	0	1.0	0	1	0	0.0
…	…	…	…	…	…	…	…	…
	29	**1**	**2**	**0.96/0.93**	**31**	**17**	**0**	**0.64/1**

H1: results

Benchmark	BayesDroid				TaintDroid			
	TPs	FPs	FNs	acc	TPs	FPs	FNs	acc
ActivityCommunication1	1	0	0	1.0	1	0	0	1.0
ActivityLifecycle1								
AnonymousClass1	0	0	0	1.0	0	1	0	0.0
…	…	…	…	…	…	…	…	…
	29	**1**	**2**	**0.96/0.93**	**31**	**17**	**0**	**0.64/1**

H1: results

Benchmark	BayesDroid				TaintDroid			
	TPs	FPs	FNs	acc	TPs	FPs	FNs	acc

```
Integer[] numbers = new Integer[]
{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,…};
      char[] imeiAsChar = imei.toCharArray();
      char[] newOldIMEI = new char[imeiAsChar.length];
      for(int i = 0; i < imeiAsChar.length; i++)
      { newOldIMEI[i] =
         Character.forDigit(numbers[(int)imeiAsChar[i]], 10); }
```

…	…	…	…	…	…	…	…	…
	29	**1**	**2**	**0.96/0.93**	**31**	**17**	**0**	**0.64/1**

H2: relaxation

relaxed detection of relevant values is effective: few false positives are introduced, and applicability to real-world apps improves

H2: methodology

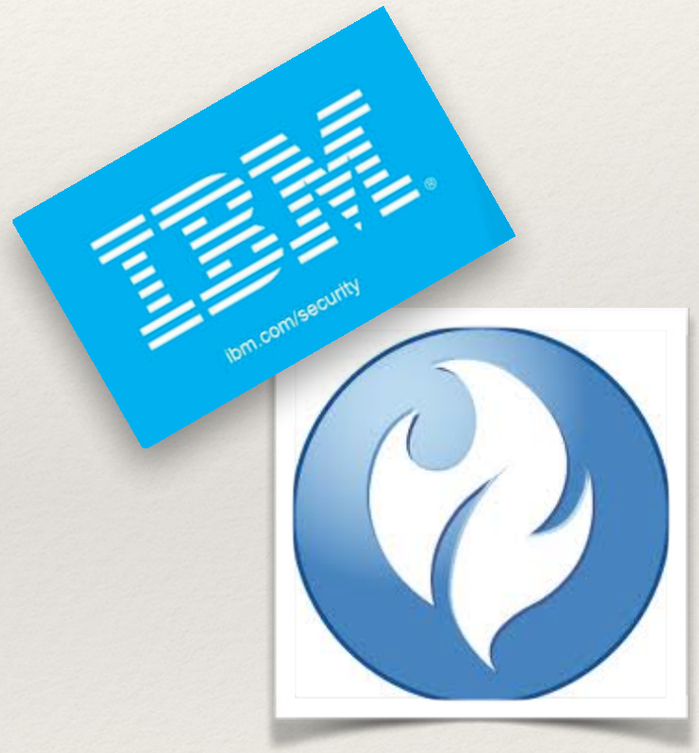
- two variants:
  - T-BD — relevant values detected via tag propagation
  - H-BD — exhaustive consideration of all values in heap graph up to a fixed bound of k=3
- benchmarks: 54 of the Google Play apps
  - subset of 65/100 apps not sampled out for training
  - excluded 8 apps w/o permission to access sensitive data and 3 that failed to install properly

H2: results

Benchmark	domain	H-BD			T-BD		
		no.	dev. ID	loc.	no.	dev. ID	loc.
atsoft.games.smgame	arcade		✓	✓		✓	✓
com.antivirus	communication		✓			✓	
com.appershopper.ios7lockscreen	personalization	✓	✓	✓			
…	…	…	…	…	…	…	…
15		**1**	**13**	**4**	**0**	**4**	**4**

conclusion

- bayesian generalization of taint-based privacy enforcement toward better precision
- value-based similarity features capture fuzziness usefully
- lower overhead and greater robustness by relaxing taint tracking



thank you / questions?

Pr(decent answer | question) =

Pr(decent answer | covered by (backup) slides)

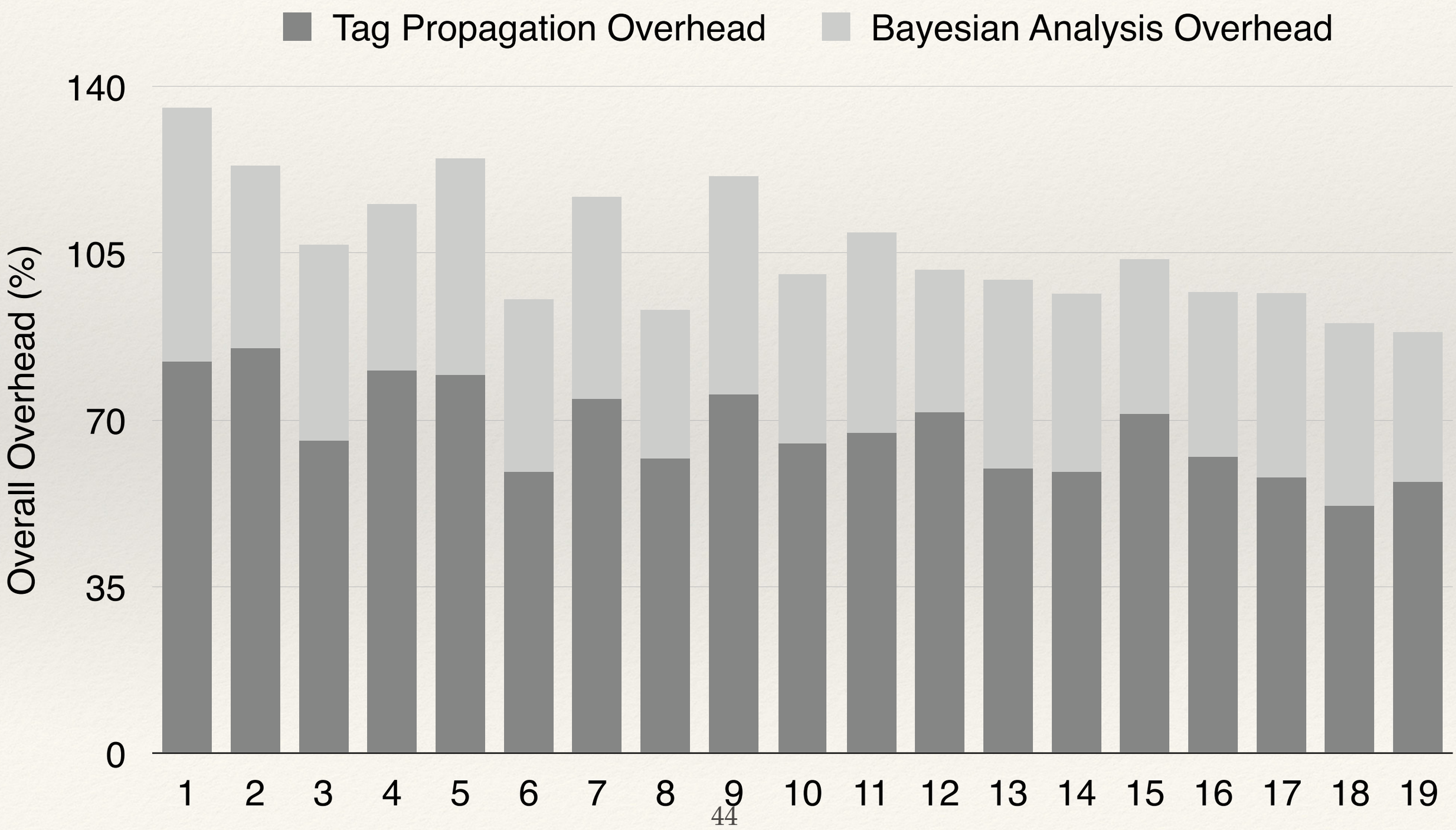
+

Pr(decent answer | 

is a decent answer)

backup

overhead


Tag Propagation Overhead
Bayesian Analysis Overhead
Overall Overhead (%)
140
105
70
35
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

false negatives

```
public class ImplicitFlow1 extends Activity {

     protected void onCreate(Bundle savedInstanceState) {
       TelephonyManager telephonyManager = (TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);
          String imei = telephonyManager.getDeviceId(); //source
          String obfuscatedIMEI = obfuscateIMEI(imei);
          writeToLog(obfuscatedIMEI);
          obfuscatedIMEI = reallyHardObfuscatedIMEI(imei);
          writeToLog(obfuscatedIMEI); }

     private String obfuscateIMEI(String imei){
          String result = "";
          for(char c : imei.toCharArray()){
               switch(c){
                    case '0' : result += 'a'; break;
                    case '1' : result += 'b'; break;
                    case '2' : result += 'c'; break;
                    …
                    default : System.err.println("Problem in obfuscateIMEI for character: " + c); } }
          return result; }

private String reallyHardObfuscatedIMEI(String imei){
     //ASCII values for integer: 48-57
     Integer[] numbers = new Integer[]{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,…};
     char[] imeiAsChar = imei.toCharArray();
     char[] newOldIMEI = new char[imeiAsChar.length];
     for(int i = 0; i < imeiAsChar.length; i++){
          newOldIMEI[i] = Character.forDigit(numbers[(int)imeiAsChar[i]], 10);}
     return newOldIMEI.toString(); }

private void writeToLog(String message){
     SmsManager sms = SmsManager.getDefault();
   sms.sendTextMessage("+49 1234", null, message, null, null); //sink leak } }
```