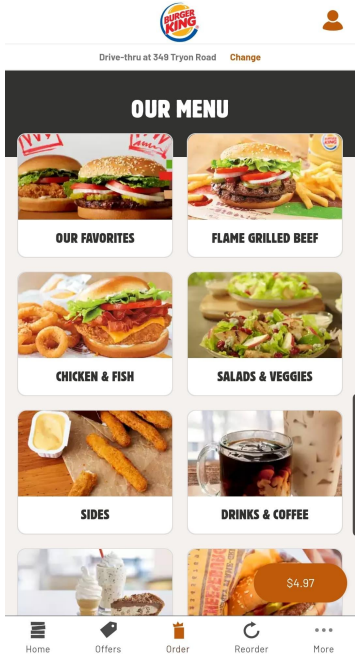
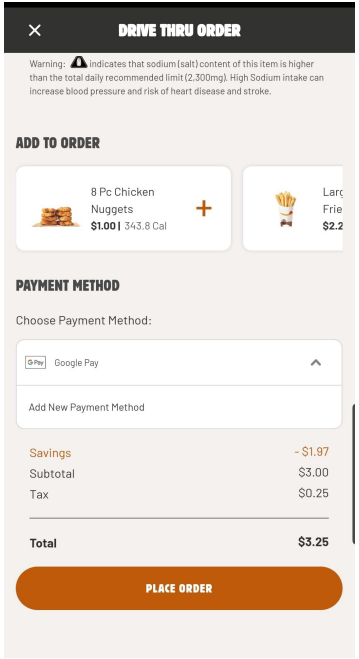
Cardpliance: PCI DSS compliance of Android applications

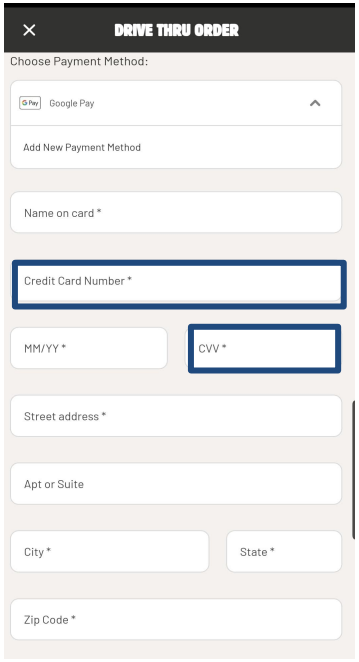
Samin Yaseer Mahmud[1], Akhil Acharya[1], Benjamin Andow[2], William Enck[1], and Bradley Reaves[1]

North Carolina State University[1]
IBM T.J Watson Research Center[2]

It All Began When Ordering a Burger!



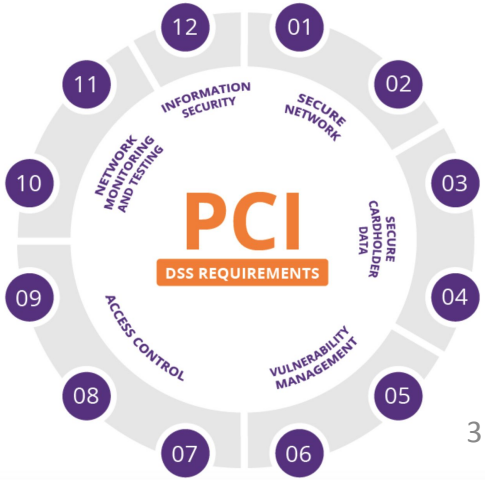




Benjamin Andow, Akhil Acharya, Dengfeng Li, William Enck, Kapil Singh, and Tao Xie, **UiRef: Analysis of Sensitive User Inputs in Android Applications**, in *Proceedings of the ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2017.

What is PCI DSS?

- Payment Card Industry's Data Security Standard that aims to secure credit card transactions against data theft or fraud
- Regulated by PCI SSC, formed in 2004 by major credit card brands
- Non-compliance comes with penalty
- PCI DSS enlists a broad set of requirements
- But not all requirements are applicable to mobile apps



PCI Security Standards Council

PCI DSS in a Mobile Context

PCI Requirements relevant to Mobile:

1. Limit CHD storage (PCI DSS requirement 3.1)
2. Restrict SAD storage (PCI DSS requirement 3.2)
3. Mask PAN when displaying (PCI DSS requirement 3.3)
4. Encrypt CHD when storing (PCI DSS requirement 3.4)
5. User secure communication (PCI DSS requirement 4.1)
6. Secure transmission of PAN to external apps (PCI DSS requirement 4.2)

Cardholder Data (CHD): Information that represents the cardholder (credit card number a.k.a PAN plus cardholder name, expiry date etc)

Sensitive Authentication Data (SAD): Information that authenticates the cardholder (PIN/ CVC/ CVV/ CVV2)

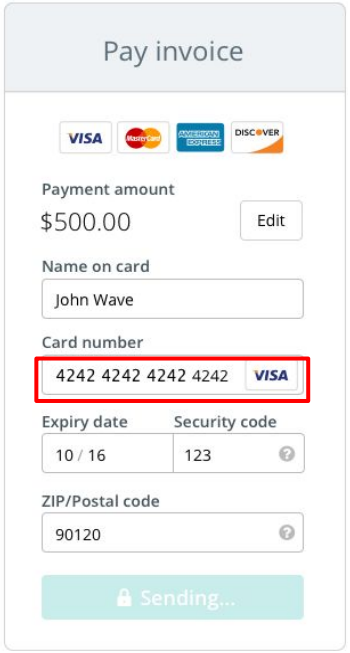
Goal and Key Challenges

Goal: To determine if Android applications are handling credit card data properly and complying to industry standards (PCI DSS).

Technical Challenges:

- Identify PCI DSS requirements relevant to mobile
- Model imprecise high level PCI requirements to static program analysis tasks
- No well defined API for taking credit card number as input
- Validate the findings

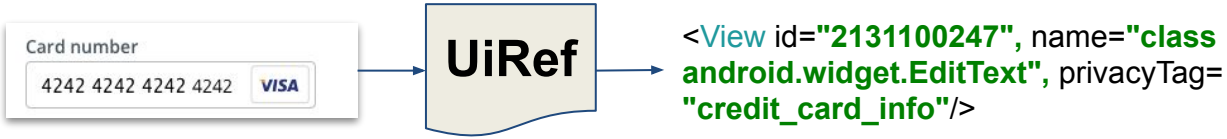
Resolving Input Semantics



How do we get information from UI?

```
EditText et = (EditText) findViewById(R.id.editText);
String creditCardNumber = et.getText().toString();
```

How do we determine it is a credit card data?



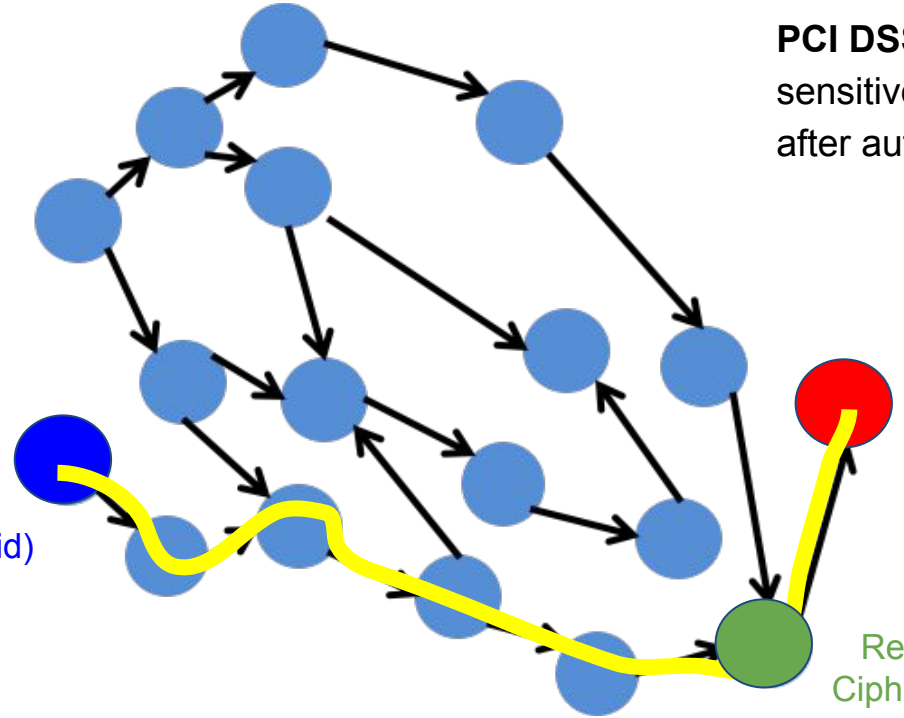
Check this value during static analysis!

```
EditText et = (EditText) findViewById( 231100247 );
String creditCardNumber = et.getText().toString();
```

Dataflow Analysis Using Cardpliance

PCI DSS 3.4 Render PAN unreadable anywhere it is stored

PCI DSS 3.2 Do not store sensitive authentication data after authorization



PCI DSS-Related Dataflow Analysis Tests

Test	Identifies	Source (S)	Sink(K)	Required (R)
T1	Storing CHD	`Activity.findViewById(ID_CC)`	DPM	-
T2	Storing SAD	`Activity.findViewById(ID_CVC)`	DPM	-
T3	Not masking PAN	`Activity.findViewById(ID_CC)`, `URLConnection.getInputStream()`	`View.setText()`	PMM
T4	Storing non-obfuscated PAN	`Activity.findViewById(ID_CC)`	DPM	OM
T5	Insecure transmission	`Activity.findViewById(ID_CC)`	`OutputStreamWriter.write()`, `OutputStream.write()`	-
T6	Sharing non-obfuscated PAN	`Activity.findViewById(ID_CC)`	`Intent.putExtra()`, `SmsManager.sendTextMessage()`	OM

Data Persistence Methods (DPM) = `java.io.OutputStream.write()`, `java.io.FileOutputSream.write()`, `java.io.Writer.write()`, `java.lang.System.out.println()`, `android.content .SharedPreferencesEditor.putString()`, `android.util.Log.i()`, `android.util.Log.d()`

PAN Masking Methods (PMM) = `java.lang.String.replace()`, `java.lang.String.substring()`, `java.lang.String.concat()`, `java.lang.StringBuilder.append()`

Obfuscation Methods (OM) = `javax.crypto.Cipher.update()`, `javax.crypto.Cipher.updateAAD()`, `javax.crypto.Cipher.doFinal()`, `java.security.MessageDigest.digest()`, `java.security.MessageDigest.update()`

Dataset Collection

Initial Data Set

- Top 500 popular apps in the 35 categories in Google Play
- Key Intuition: *Popular apps have greater impact*

Filtering Non-relevant Apps

- Most apps *do not* handle credit card information
- We filtered apps by searching resource files for credit card-related search strings
- 1,868 applications had such matches

Final Data Set

- Amandroid was able to generate data dependency graph in **358** matching apps

Application Study

Cardpliance Analysis

- We ran our 6 PCI checks on 358 applications
- Cardpliance reported 20 applications violated at least one PCI check
- Another 20 applications have bad `SocketFactory` classes

Manual Validation

- We manually validated 40 apps over a one month period with the JEB decompiler
- We confirmed **15 PCI violations across 6 applications**

App Name	Downloads	T1	T2	T3	T4
Credit Card Reader	500K+	✗			✗
Fast Toll Illinois	10K+	✗	✗		✗
Bens Soft Pretzels	10K+	✗	✗	✗	✗
The Toll Roads	100K+	✗	✗		✗
Connect Network by GTL	1M+			✗	
Peach Pass GO!	50K+	✗			✗

Highlighted Findings

The Good:

- Most of the popular applications are doing it right. We find **98.32%** of the 358 applications were likely PCI DSS-compliant
- Applications **are correctly performing hostname and certificate verification** when sending payment information over SSL connections.
- Applications are **not sending** credit card numbers to insecure HTTP URLs
- Applications are **not insecurely sharing payment information** via SMS or ICC channels

Highlighted Findings

The Bad

- 6 applications were not PCI DSS compliant
- Applications **log sensitive information** like credit card numbers and CVC

The Ugly:

- More than **1.5 million** users impacted by the non-compliant apps

Disclosure:

- We disclosed findings to the application developers in November 2019
- 1 out of 6 developers responded (16.6%)

Case Studies

Credit Card Reader

Merchant Account Solutions Business

Everyone

You don't have any devices.

Add to Wishlist

Updated	Size	Installs
June 28, 2020	5.9M	500,000+
Current Version	**Requires Android**	**Content Rating**
20.06.26	5.0 and up	Everyone Learn More

Logging customer's credit card number:

```
@Override  // android.view.View$OnClickListener
public void onClick(View v) {
    switch(v.getId()) {
        case 0x7F060002: {  // id:action_next
            Intent i = new Intent(this, TipActivity.class);
            if(this.cc_sales_tax.isChecked()) {
                i.putExtra("sale_amount", String.format("%.2f", Double.val
            }
            else {
                i.putExtra("sale_amount", this.sale_amt);
            }

            i.putExtra("cc_no", this.cc_no.getText().toString());
            i.putExtra("cc_exp", this.cc_exp.getText().toString());
            i.putExtra("cc_cvv2", this.cc_cvv2.getText().toString());
            i.putExtra("cc_zip", this.cc_zip.getText().toString());
            i.putExtra("cc_st_add", this.cc_st_add.getText().toString());
            this.startActivity(i);
            break;
        }
    }

    Log.d("CCR - Payment", this.cc_no.getText().toString());
}
```

Case Studies

Bens Soft Pretzels

RT7 Inc Food & Drink

Everyone

You don't have any devices.

Add to Wishlist

Logging Credit Card Number and CVC:

```
this.ReloadClick.setOnClickListener(new View.OnClickListener() {
    @Override  // android.view.View$OnClickListener
    public void onClick(View arg15) {
        Log.d("Click", "Click");
        Log.d("creditCardNumber", ReloadPage.creditCardNumber);
        Log.d("cvccc", ReloadPage.cvccc);
        double v2 = 0;
        if(ReloadPage.amountToReload.doubleValue() == v2) {
            ReloadPage.this.showalert( arg5: " ",  arg6: "Reload Amount");
        }
```

Credit Card number as Shared Preference Key:

```
this.cardForm.setOnCardTypeChangedListener(new OnCardTypeChangedListener() {
    @Override
    public void onCardTypeChanged(CardType arg3) {
        CreditCardEnterPage.cardname = arg3.name();
        Log.d("CardName", CreditCardEnterPage.cardname);
        SharedPreferences.Editor v3 = PreferenceManager.
                getDefaultSharedPreferences(CreditCardEnterPage.
                        this.getApplicationContext()).edit();
        v3.putString
                ("GetDataCardName".concat(CreditCardEnterPage.userId).
                concat(CreditCardEnterPage.userCardNumber),
                        CreditCardEnterPage.cardname);
        v3.apply();
```

Vulnerable implementation of SecureRandom object:

```
private SecretKeySpec setthekey() {
    SecretKeySpec v0_1;
    try {
        SecureRandom v0 = SecureRandom.getInstance("SHA1PRNG");
        v0.setSeed(CreditCardEnterPage.userId.
                concat(CreditCardEnterPage.userCardNumber).getBytes());
        KeyGenerator v1 = KeyGenerator.getInstance("AES");
        v1.init(0x80, v0);
        v0_1 = new SecretKeySpec(v1.generateKey().getEncoded(), "AES");
    }
    catch(Exception unused_ex) {
        Log.e("AES Error", "AES secret key spec error");
        v0_1 = null;
    }

    if(v0_1 != null) {
        String v1_1 = Base64.encodeToString(v0_1.getEncoded(), 0);
        SharedPreferences.Editor v2 = PreferenceManager.
                getDefaultSharedPreferences(this.getApplicationContext()).edit();
        v2.putString("GetDataPoss".concat(CreditCardEnterPage.userId).
                concat(CreditCardEnterPage.userCardNumber), v1_1);
        Log.d("ToChangedStores", v1_1);
        v2.apply();
    }

    return v0_1;
}
```

Limitations

Inherent to Underlying tools:

- Over approximation of Amandroid (false positives)
- Context insensitive analysis of bad `SocketFactory` and `TrustManager`
- Imprecision in UiRef

Limitation of our Approach:

- Keyword based search introducing false negatives
- No analysis of weak cipher suite or hard coded key
- Lightweight heuristic on test T3

Summary

- Cardpliance performs PCI DSS compliance checks on Android applications.
- The landscape of **popular** Android applications in terms of PCI DSS is fairly positive.
- Source code available (https://github.com/wspr-ncsu/cardpliance)

Samin Yaseer Mahmud
Wolfpack Security and Privacy Research Lab
Department of Computer Science
North Carolina State University
(smahmud@ncsu.edu)

Team Members
William Enck (NC State)
Bradley Reaves (NC State)
Benjamin Andow (IBM)
Akhil Acharya (NC State)