UI and Security: Two Sides of the Story

Wenke Lee
Georgia Institute of Technology

Who Cares About UI?

- Users do!
  - That is how they use computers
- "Users are the weakest link in security"
- So, let's look at the interplay between UI and security

UI Can Both Hurt and Help Security

- New UI features can introduce new vulnerabilities
  - Not just in the UI code but also in the underlying system

- UI can be leveraged to improve security
  - Infer user intent and apply the appropriate security policy

The Insecurity Because of UI

- Accessibility introduces new paths of privileged access to system resources

- Powerful window manipulation permission with accessibility on Android can hide attacks from users

Computer Accessibility (a11y)

- For the person with disabilities
  - –Visually impaired
    - •Text-to-Speech reader
  - –Hearing impaired
    - •Captioning service
  - –Motor impaired
    - •Voice Commander
  - –Keyboard impaired
    - •On-screen keyboard

Accessibility Library

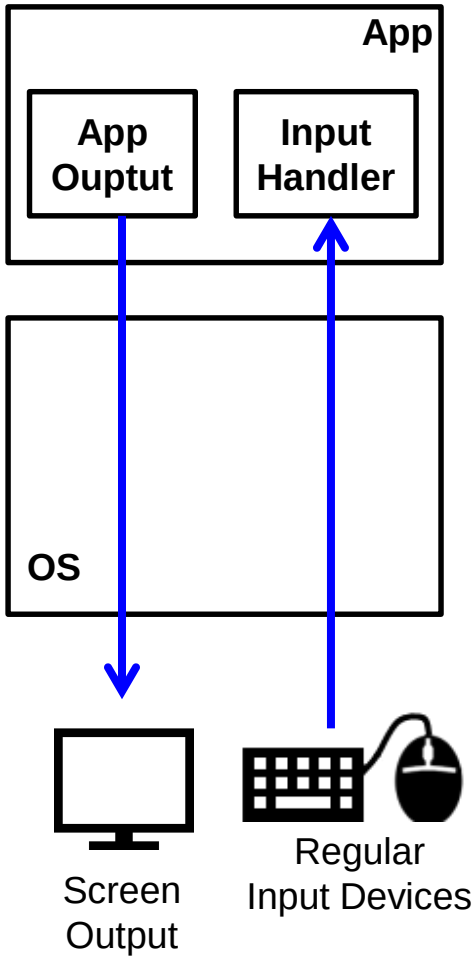
- OS opens API for developing A11y features
  - –Available in Windows, OS X, Ubuntu, iOS, Android, etc.
- Capabilities
  - –Read UI states of the system
  - –Perform actions on UI elements
    - •Click
    - •settext ()
    - •etc.

Security Implications of A11y

- Creates new I/O Paths

- Break basic/traditional assumptions on I/O
  - –Input comes from the user
    - •Through a11y interface, a program can send input event to the application
  - –Output can only be seen by the user
    - •A11y interface allows to a program can read output of the other applications

Traditional I/O Paths in OS



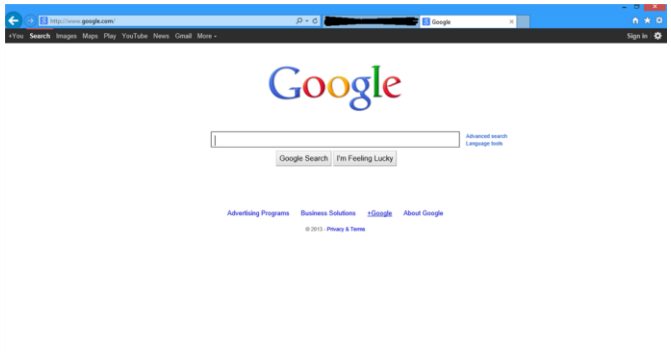
A11y Added New I/O Paths to OS

1. User Speaks

2. Voice commander translate it into machine command

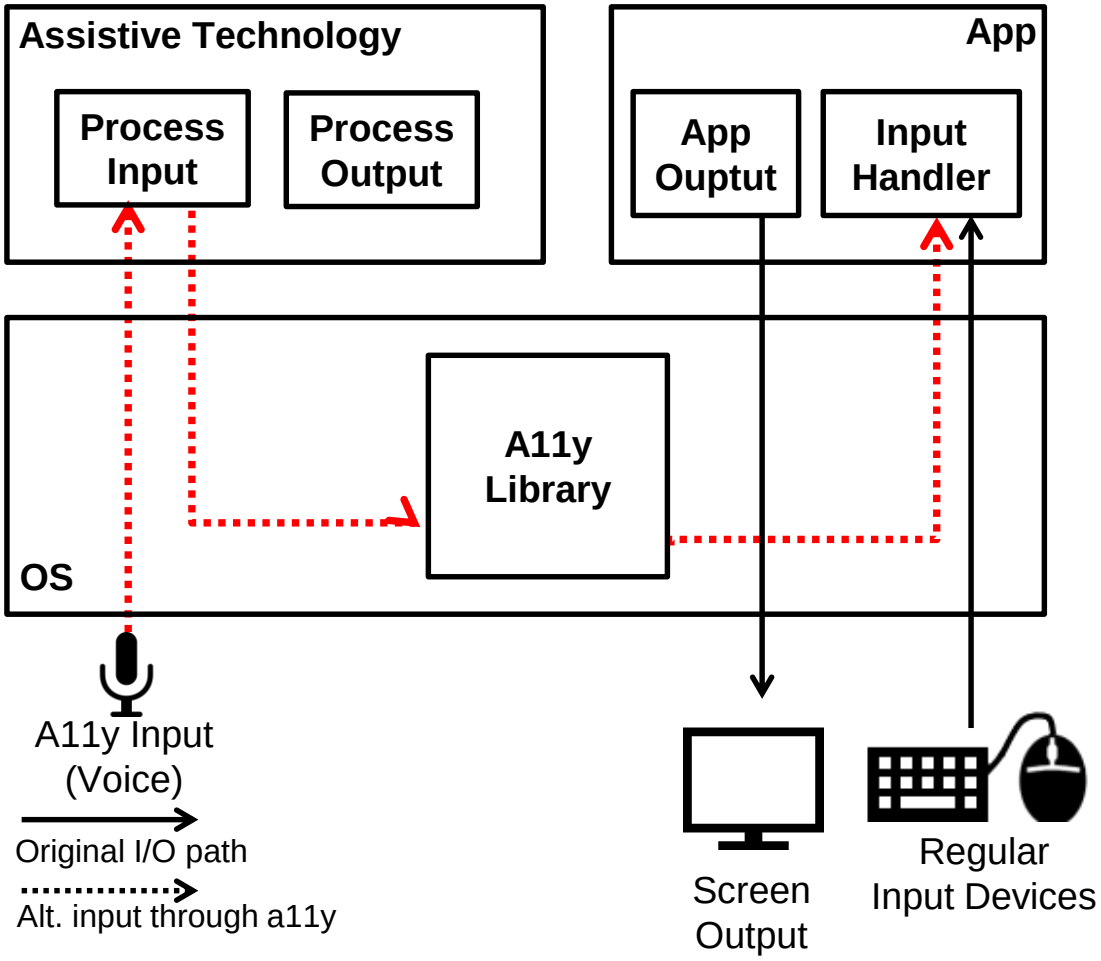
1) Click address bar
2) Type google.com

3. OS delivers command to the app (a11y library)



4. App is controlled by Voice

A11y Added New I/O Paths to OS



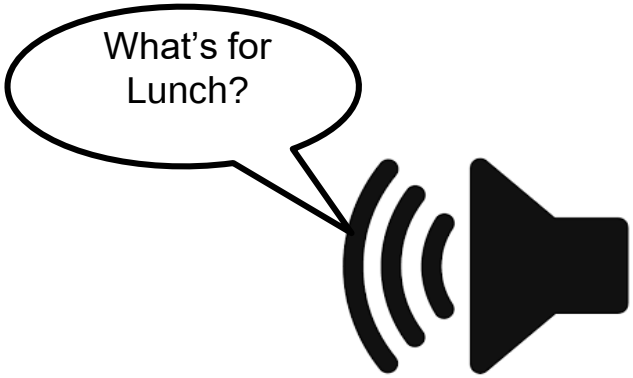
A11y Added New I/O Paths to OS

1. App shows text output

Prepare audio for text…

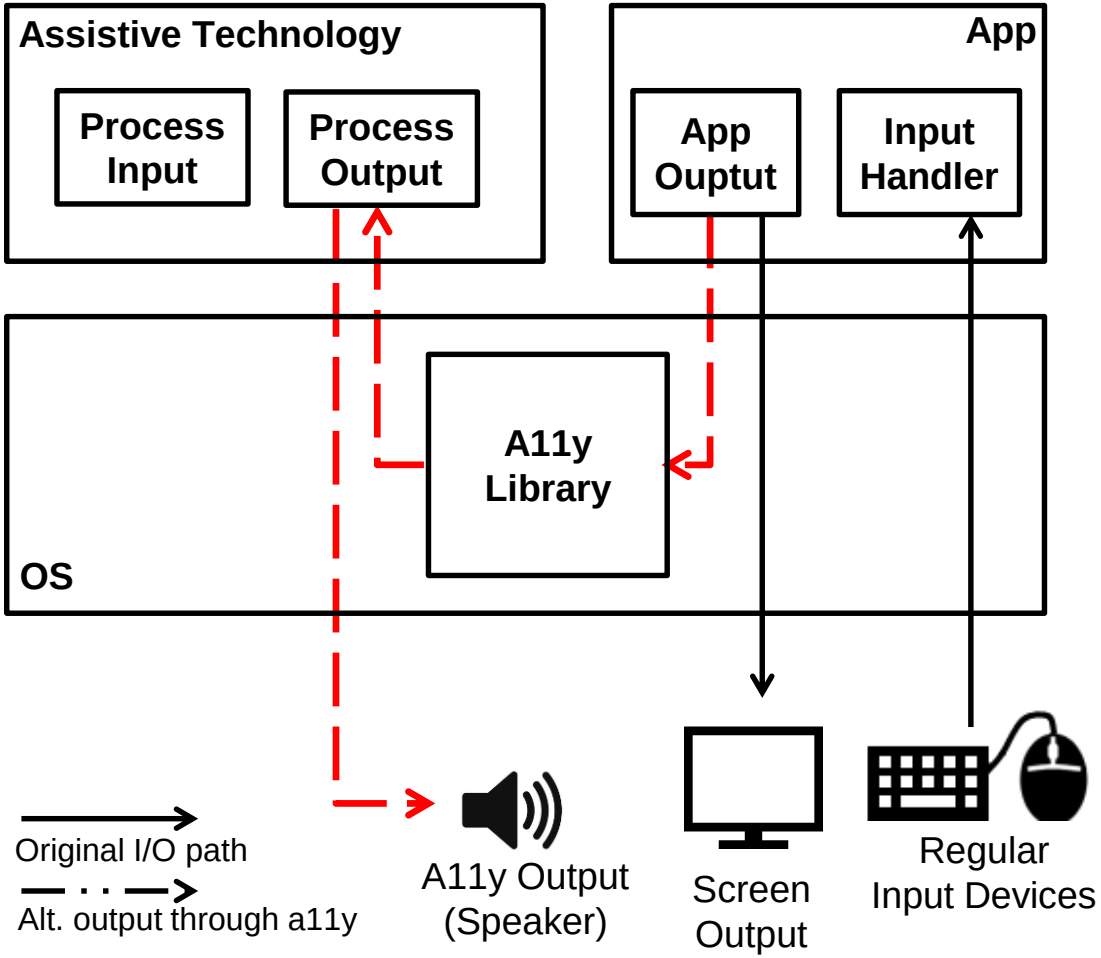
2. Screen Reader gets app output

3. OS gets audio playing request



4. User receives audible output

A11y Added New I/O Paths to OS



Required Security Checks

- Security checks must be put in the right places
  - –Does a11y input really comes from the user?
- Checks can be placed in three different level
  - –Assistive Technology (processor of alternative I/O)
  - –Operating System
  - –Application (protect themselves from alternative I/O)

At Assistive Technology (AT) Level

Required checks at AT level

Is the voice from real human?

If not, machine can access it!

Is the voice matched with registered user?

If not, any other human user can access it!

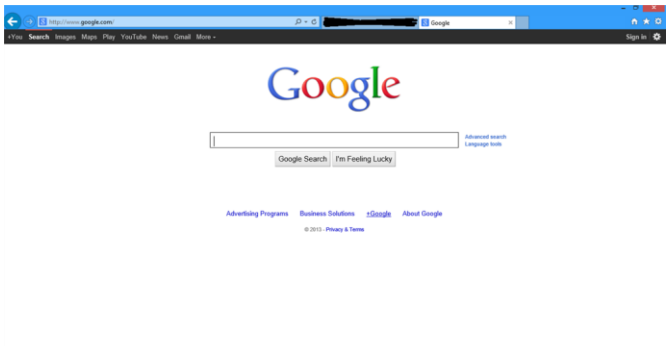
At OS Level

Required checks at OS level

Is this assistive technology allowed to access a11y?

If not, any program (possibly malware) can access it!

At Application Level



Required checks at application level

Should I react to input from a11y features?

In particular for security sensitive UI actions!

Evaluating A11y Security in OSes

- Objective
  - –Check if OSs are secure under attacks through new I/O paths created by supporting A11y
- Method
  - –Analyze OS for accessibility features
    - •Programmatic access to I/O event
    - •Voice commander, password viewer, etc.
  - –Test existence of required security checks
    - •If not, try to launch an attack

Evaluating A11y Security in OSes

- Target
  - 4 Major OSes
    - MS Windows 8.1, Ubuntu 14.04 Linux
    - iOS 6, and Android 4.4
- Focus
  - Try to evaluate OS default settings
    - AT-level check
      - Voice Commander
    - OS-level check
      - Programmatically controllable I/O
    - App-level check
      - We do not perform the evaluation …

Evaluation on A11y Input

Platform	AT-level check (voice commander)	OS-level Security Check	Vulnerable?
Windows	None (Speech Recognition)	UIPI	YES
Ubuntu	N/A	None	YES
iOS 6	None (Siri)	None	YES
Android	Voice Authentication (Moto X)	User Settings Required	YES

Evaluation on A11y Output

Platform	Reading of UI Structure	A11y leaks on screenshot	Password protection	Vulnerable?
Windows	UIPI	Yes	Yes	YES
Ubuntu	None	No	Yes, but incomplete	YES
iOS 6	N/A	Yes	N/A	YES
Android	User Settings Required	No	User Settings Required	YES

Attacks for missed checkpoint

- We tried to launch attacks if any of security check is missing
- We found 12 new attacks
  - –Windows (3)
    - •2 Privilege escalation, 1 password leak
  - –Linux (2)
    - •Bypassing process boundary, password leak
  - –iOS (4)
    - •Bypassing sandbox and authentication
    - •Privilege escalation, Password leak
  - –Android (3)
    - •Bypassing sandbox and authentication
    - •password leak

Attacks on Voice Commander

- Voice commander accepts non-human voice
  - –Any app capable to play audio can send command
    - •Broken assumption: input comes from the user
  - –No authentication
    - •Windows Speech Recognition
    - •Siri
    - •Google Now
  - –Voice authentication in presence
    - •Moto X
      - –Vulnerable to replay attack

Privilege Escalation in Windows

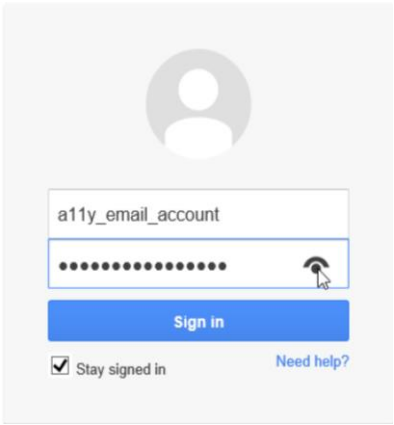
- Malware runs as normal user can execute Speech Recognition
- Speech Recognition automatically launches with administrative privilege
  - –Let A11y user control admin stuff …
- Malware can get admin privilege by sending voice command to Speech Recognition

Take Control Over Other Apps

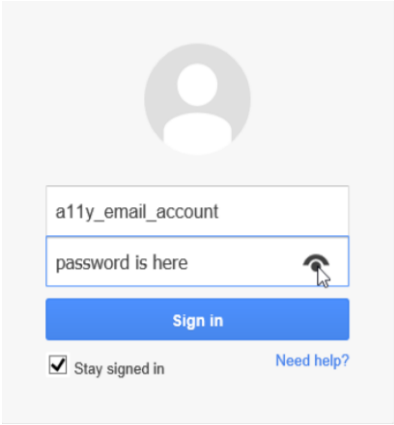
- A11y library allows a program send input to the other apps
  - –Broken assumption: input comes from the user
- Privilege escalation
  - –Windows
    - •Send click to a security-sensitive dialog
- Bypassing app sandbox
  - –iOS and Android
    - •Sending programmatic input to the target app

Stealing Password!

- Applying image processing on screenshot leaks password string.



A) Before clicking Eye



B) After clicking Eye

Discussions

- Root-cause
  - –Maximizing compatibility
    - •The UI is expected to run as if it gets the real input on a11y request
    - •Programmatic input processed as same as the real one

	Real Touch Click	A11y Click
Intermediate func	onTouchEvent()	performA11yActionInternal()
Final handler in UI	performClick()	performClick()

The same PerformClick() is called

```
// On real touch event
public boolean onTouchEvent(MotionEvent event) {
  switch (event.getAction()) {
    case MotionEvent.ACTION_UP:
    {
      // ...
      // performClick() is called to handle real click event
      performClick();
      // ...
    }
  }
}

// On a11y request for click
boolean performAccessibilityActionInternal(int action,
                                           Bundle arguments) {
  // ...
  switch (action) {
    case AccessibilityNodeInfo.ACTION_CLICK:
    {
      if (isClickable()) {
        // the same performClick() is invoked to handle a11y request
        performClick();
        return true;
      }
    } break;
  }
  // ...
}
```

Discussions

- Root-cause
  - –Problems when it handled user input and ATK input differently
    - •On gksudo dialog, copytext() works while Ctrl-C does not work!
    - •New implementation could miss security checks.

```
GTK::CopyText() {
        if(text->isVisible)
                return text
        else
                return null;
}
```

```
ATK::CopyText() {
                …
                return text
                …
}
```

Discussions

- Root-cause
  - –No correct authentication for alternative input
    - E.g., any program can send fake voice …
  - –Technical & economical difficulty
    - •Possible solution for voice authentication
      - –Liveness check
      - –Challenge-response
    - •Practical issues
      - –Processing power
      - –Power consumption
      - –Etc.

Discussions

- Root-cause
  - –Weak access control on a11y libraries
    - •Windows: None
    - •OS X : None
    - •Ubuntu: None
    - •iOS 6 : None -> patched in iOS 7
    - •Android: User settings
      - –Not enough …

Leveraging UI to Improve Data Protection?

What data is important/valuable?

Ask the user?

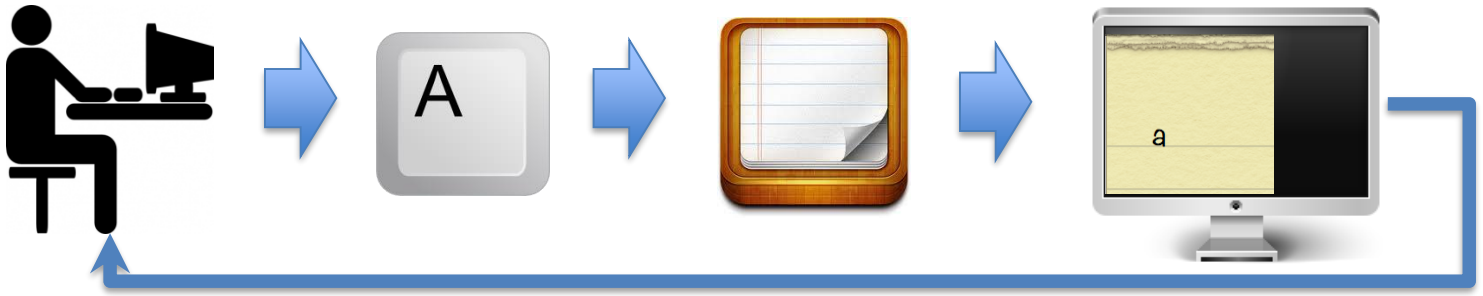
What? Aren't They the Weakest Link?

What data is important/valuable?

User-Intent Monitoring of (Text-Based) Networked Applications

User Intent from UI

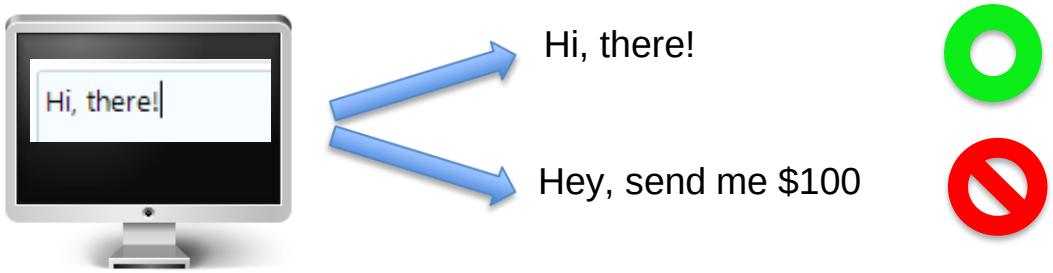
- User interacts with computer using input/output hardware
  - Input: Keyboard, Mouse
  - Output: Display screen
- Feedback loop in user interaction



Capturing User Intent

- Observation
  - –User verifies what their input by on-screen display

- A New Security Policy
  - –What You See Is What You Send (WYSIWYS)
    - •On-screen text is user-intended
    - •Only allows traffic that matches on-screen text



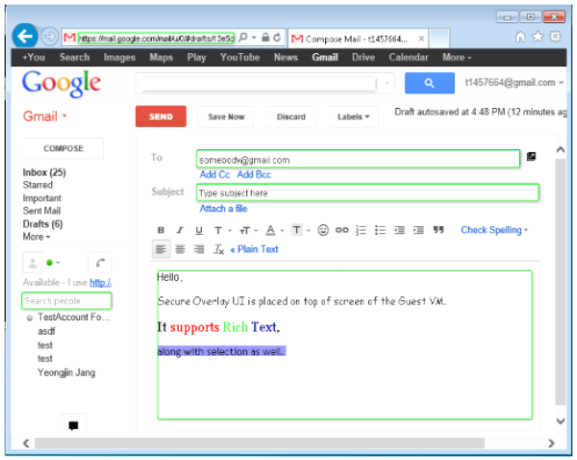
What You See Is What You Send

- WYSIWYS (Facebook example)

What You See Is What You Send

- Supposed we have a security monitor for WYSIWYS
  - –Compares outgoing traffic data with data on application GUI
  - –Needs to query application for data on its GUI
  - –If the application is already compromised
    - •It can lie about data on GUI
- Cannot trust application GUI
- The security monitor must have control over “GUI”

Security Overlay

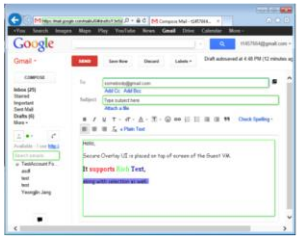


Combined Screen

On-screen text is always same with captured text on the security monitor.

Security Overlay

- Only re-draws certain elements, e.g., editbox
  - –Exactly same location, size, and color
  - –Can support rich-text
    - •Font, size, color, style, etc.

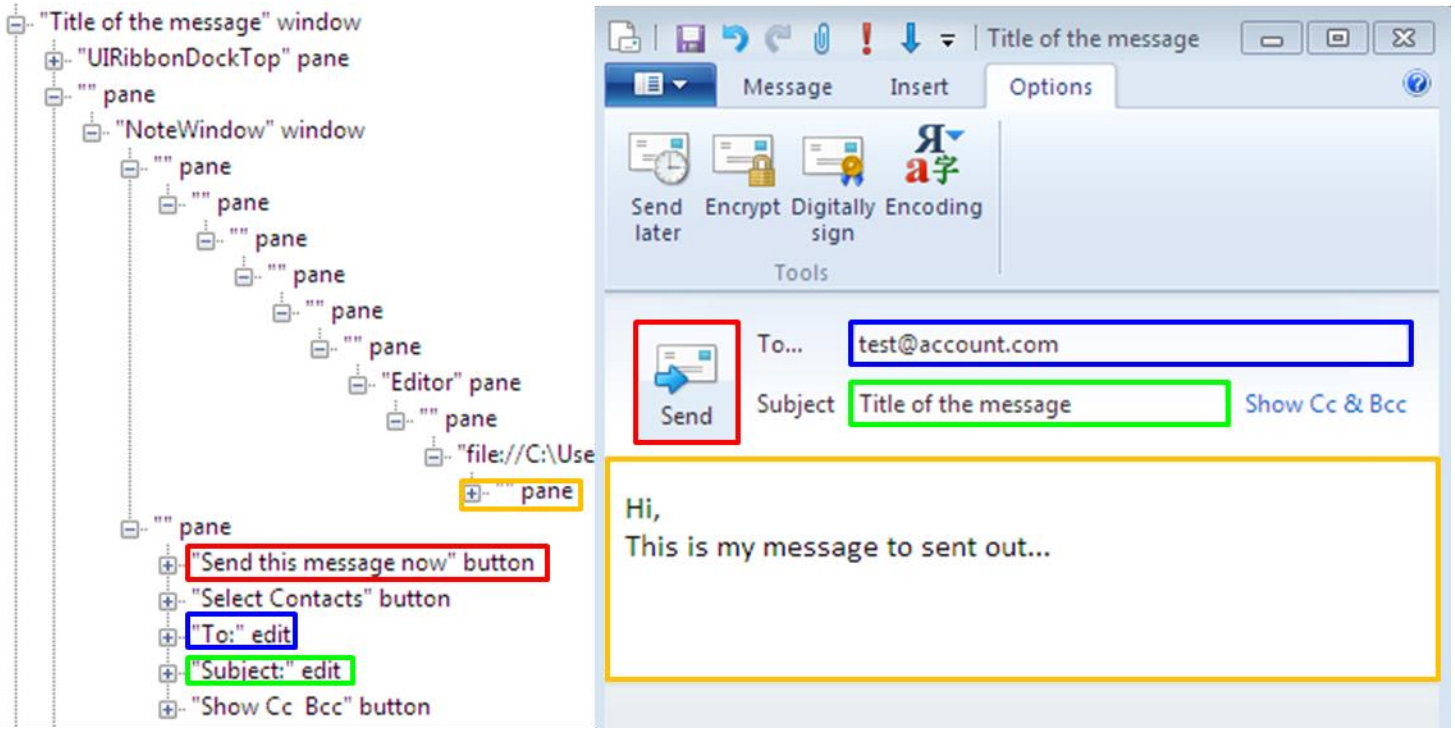


Combined Screen

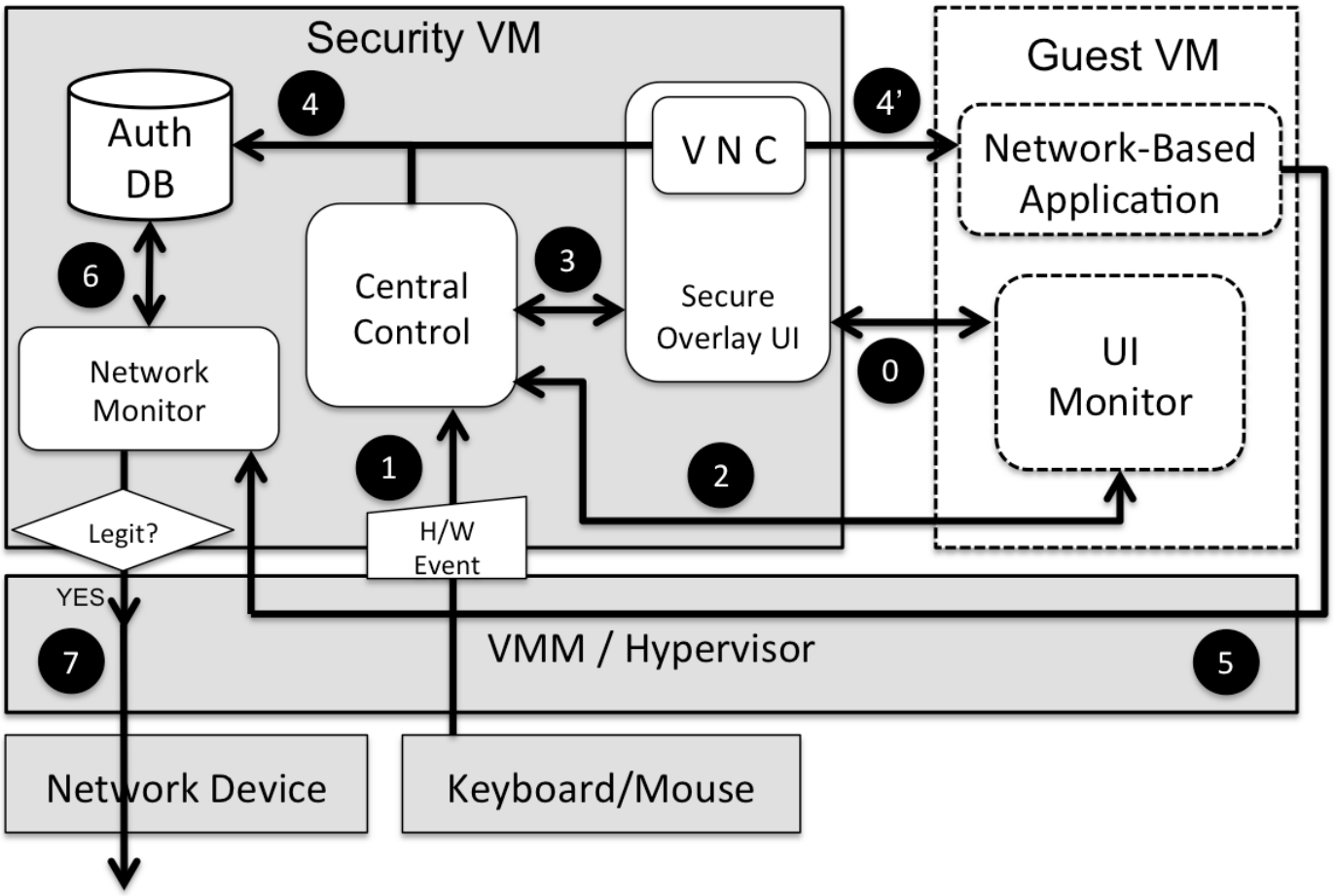
- Passive UI
  - –It does not gets any user input
  - –Content updated after each applications gets input
  - –Support selection, copy/paste, spell correction, auto-completion, etc…

UI Monitor

- Uses library for UI Testing (UIAutomation)



System Architecture and TCB



Threat Model

- Hypervisor and security VM is fully trusted
  - –Assumes VM escape is impossible

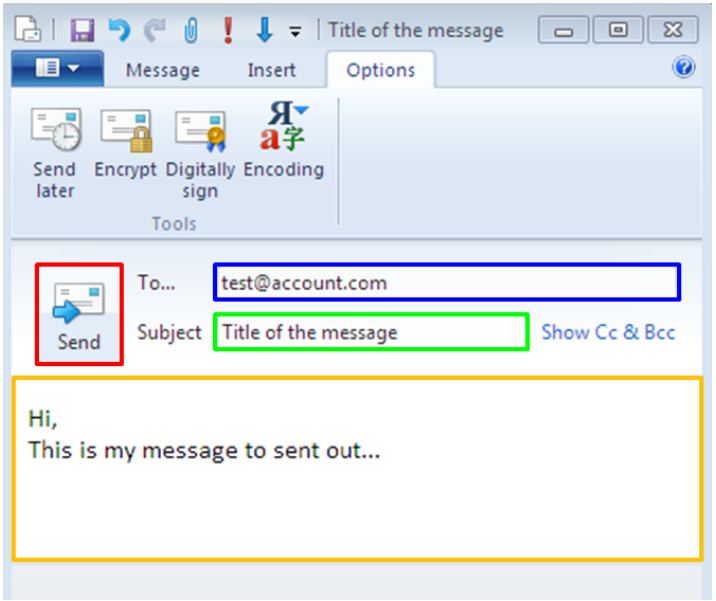
- Hardware input devices are trusted, and attacker has no physical access
  - –Attacker cannot forge hardware input event

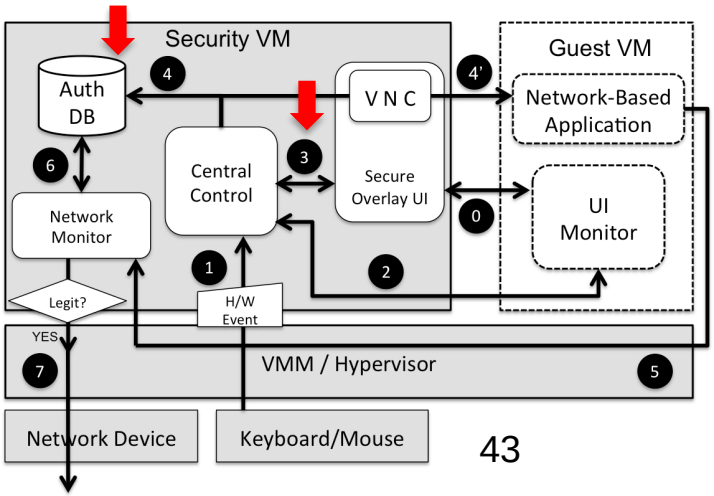
Threat Model (Cont'd)

- All hardware input event is interposed at hypervisor first, then delivered to User VM
  - –Security VM cannot miss hardware event, and User VM cannot emulate it
- Completely distrust User VM
  - –Allows all attacks including Kernel-level malware
    - •UI monitor is untrusted

Capturing User Intent

- Extract all required text from Secure Overlay when traffic-triggering event happens
  - –Store it to Authorization DB for enforcement at network level.



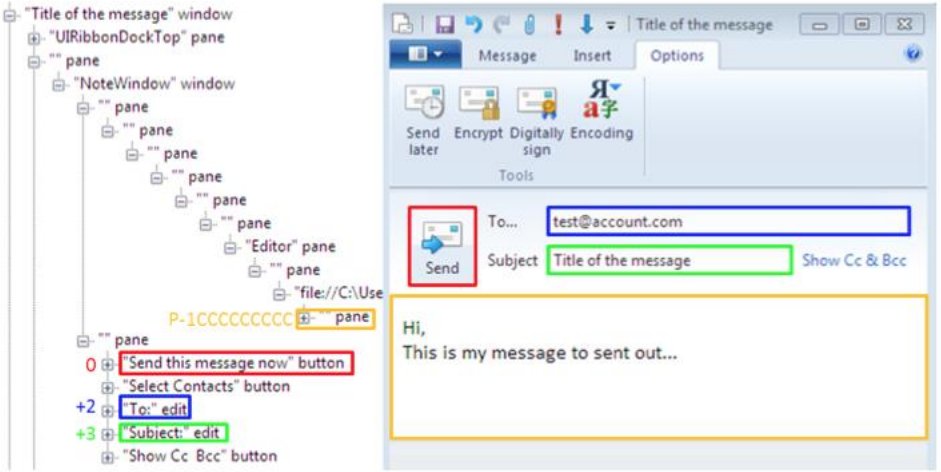


Application-Specific Logics

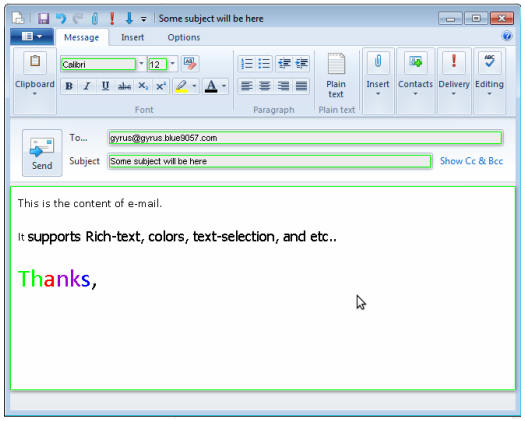
- User Intent Signature

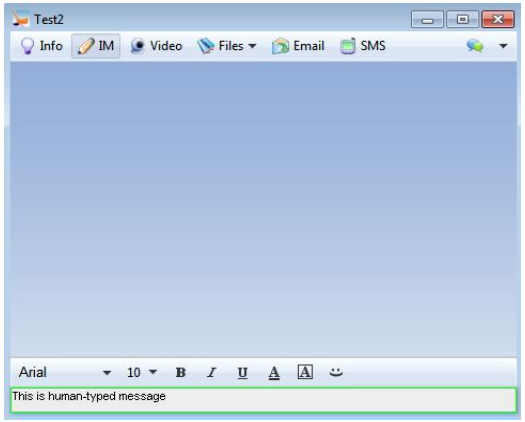
Example 1 User Intent Signature for sending e-mail on Windows Live Mail.

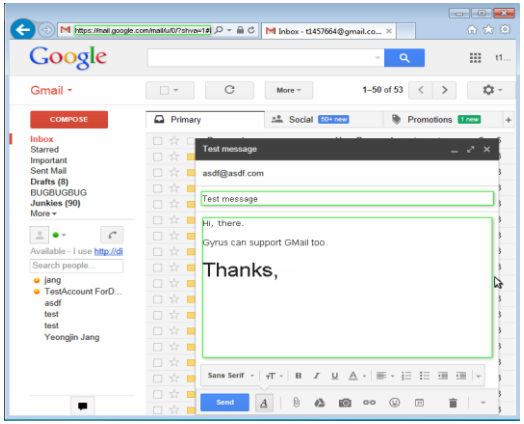
```
{
  "TAG" : "LIVEMAILCOMPOSE",
  "EVENT" : "LCLICK",
  "WINDOW" : "ATH_Note",
    "COND" : {
      "0" : {
        "CONT" : "BUTTON",
        "NAME" : "Send this message now"
      },
      "+2" : {
        "CONT" : "EDIT",
        "NAME" : "To:"
      },
      "+3" : {
        "CONT" : "EDIT",
        "NAME" : "Subject:"
      },
      "P-1CCCCCCCCC" : {
        "CONT" : "PANE"
      }
    },
    "CAPTURE" : {
      "A" : "+2.value",
      "B" : "+3.value",
      "C" : "P-1CCCCCCCCC.value"
    },
    "TYPE" : "SMTP",
    "BIND" : {
      "METHOD" : "SEND",
      "PARAMS" : {
        "to" : "A",
        "subject" : "B",
        "body" : "C"
      }
    }
}
```

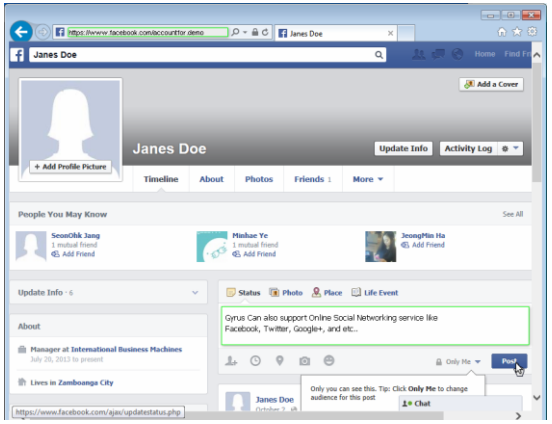


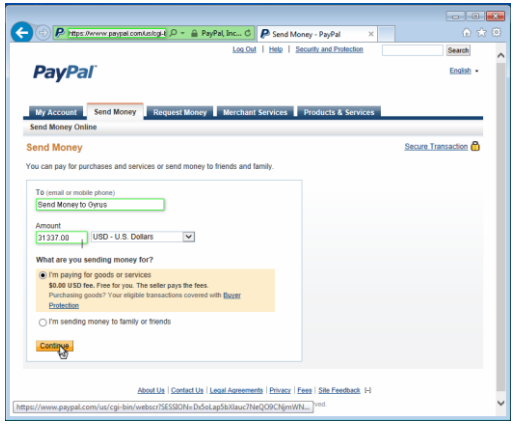
Application Examples





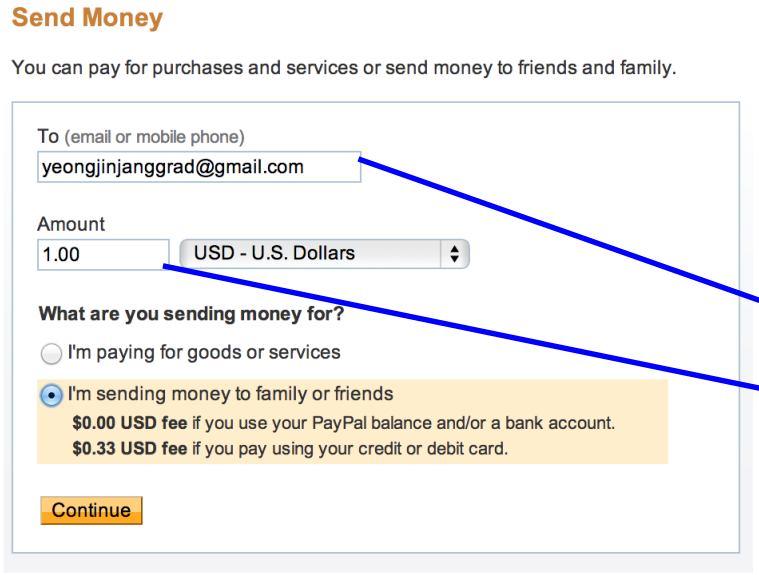


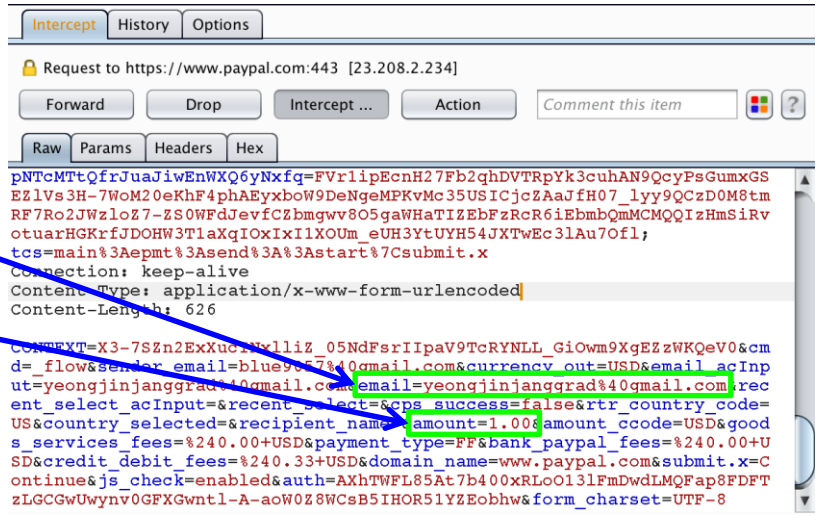




Paypal Example

- Paypal “Send Money”



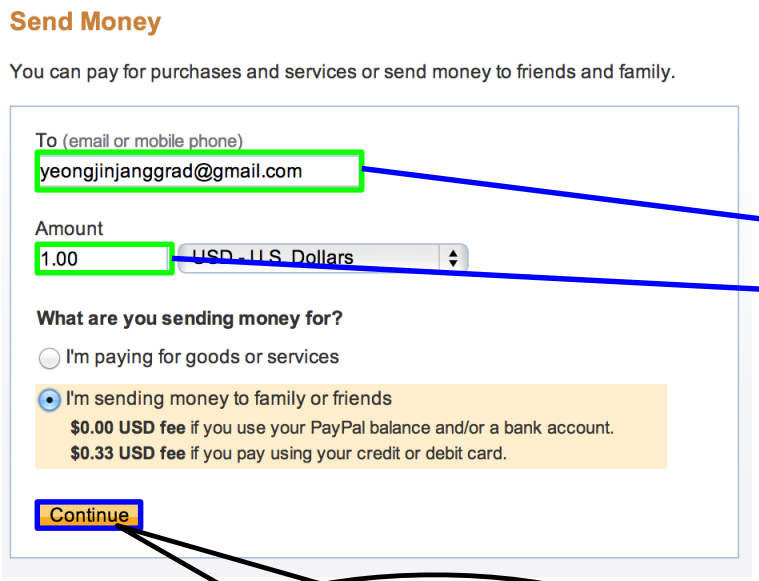


What user sees when sends money

Outgoing network traffic

Paypal Example

- Capturing User Intent



{

"ACTION" : "Paypal Send",
"Recipient" :
yeongjinjanggrad@gmail.com,
"Amount" : "1.00"

}

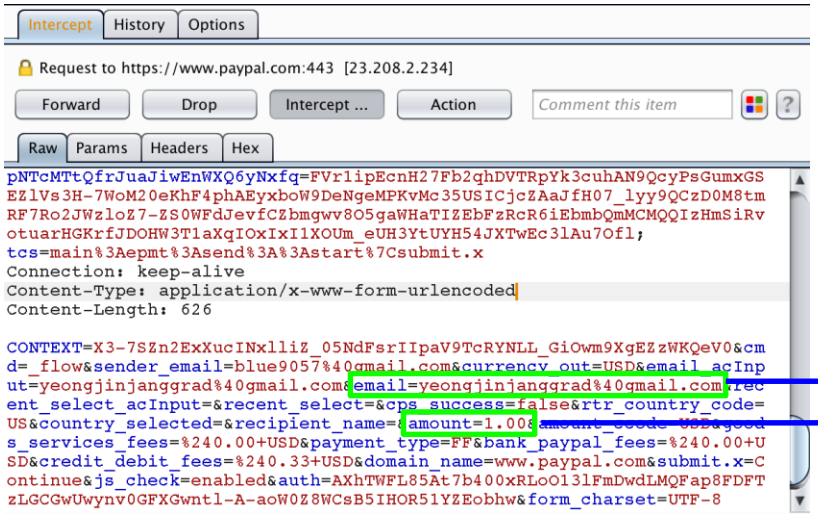
When Continue is clicked,
Stores e-mail and amount

Authorization Vector

Store it to
Authorization DB

Paypal Example

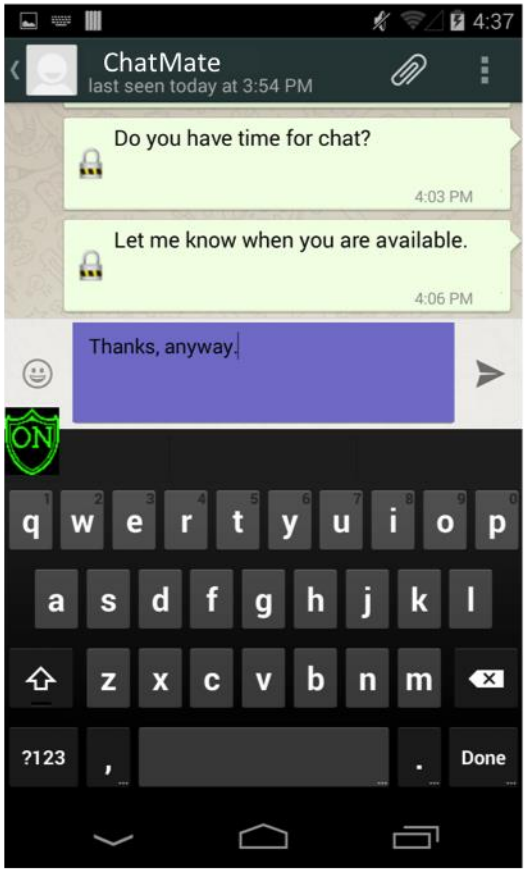
- Monitoring Network Traffic
  - –Apply Deep Packet Inspection



Passes the traffic only if it is matched with previously authorized data in DB

Authorization Vector

Security Overlay For Messaging App



What User Sees

Security Overlay

Application UI

Summary

- Security is about data protection
  - –What data is important to user?
  - –Usability is the key
    - •Do users have to change workflow?
- Security overlay
  - –A systems mechanism
  - –Monitors user intents through on-screen UI data, and security policy
    - •Integrity and confidentiality protection
  - –Transparent
  - –General
    - •Applicable to class(es) of applications

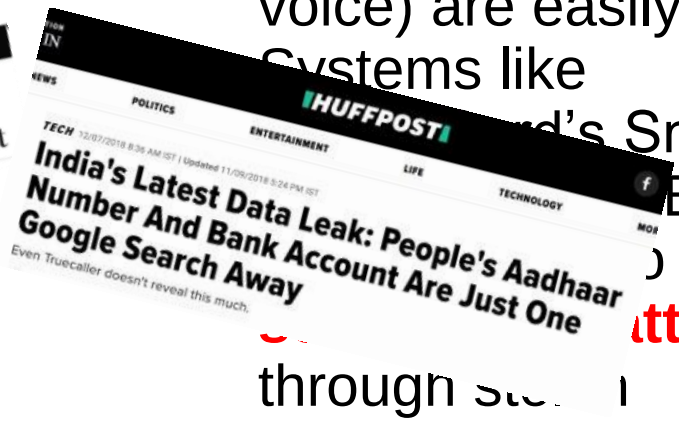
Biometric Authentication

Image Credit: IBM Future of Identity Study

Challenges

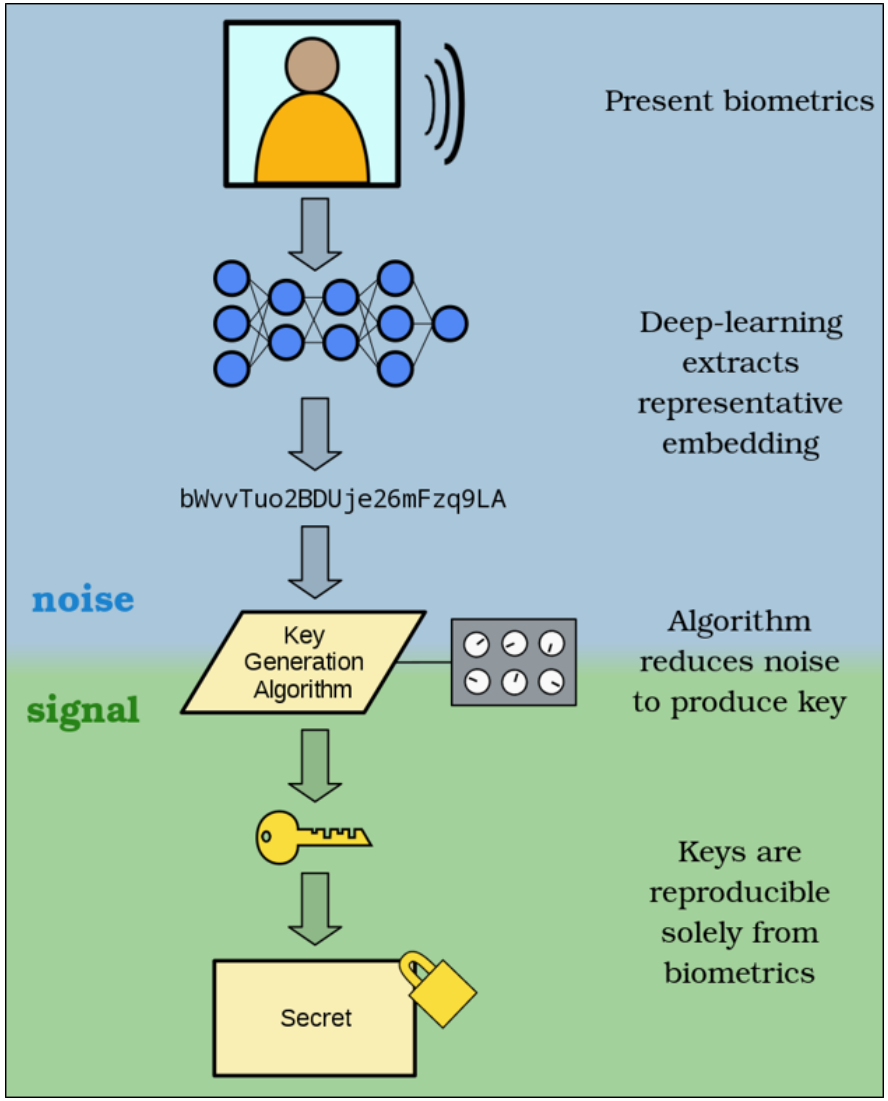
- **Privacy Issues**
  - No existing method for **remote** biometric-based authentication on **encrypted** data
  - Current practice is storing raw biometrics in the authentication server
  - ake people give their biometrics to most services

- **Stolen Biometrics (Live?)**
  - You **cannot replace** a biometric
  - Most popular biometric sources (e.g. face, voice) are easily stolen
    Systems like d's Smile-to-Blink-to-
    large-ttacks
    through st
    biometrics



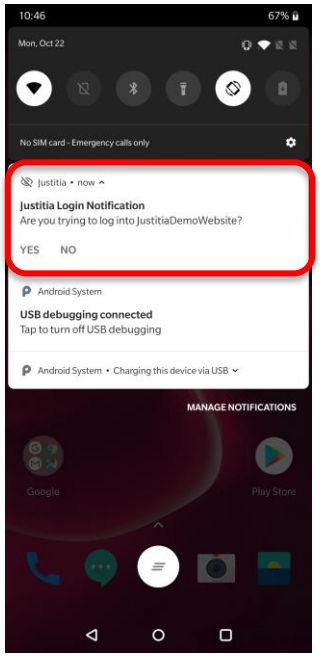
Protecting Privacy

- Allow biometric matching over **encrypted** biometric data on remote authentication server which enables **recovery**

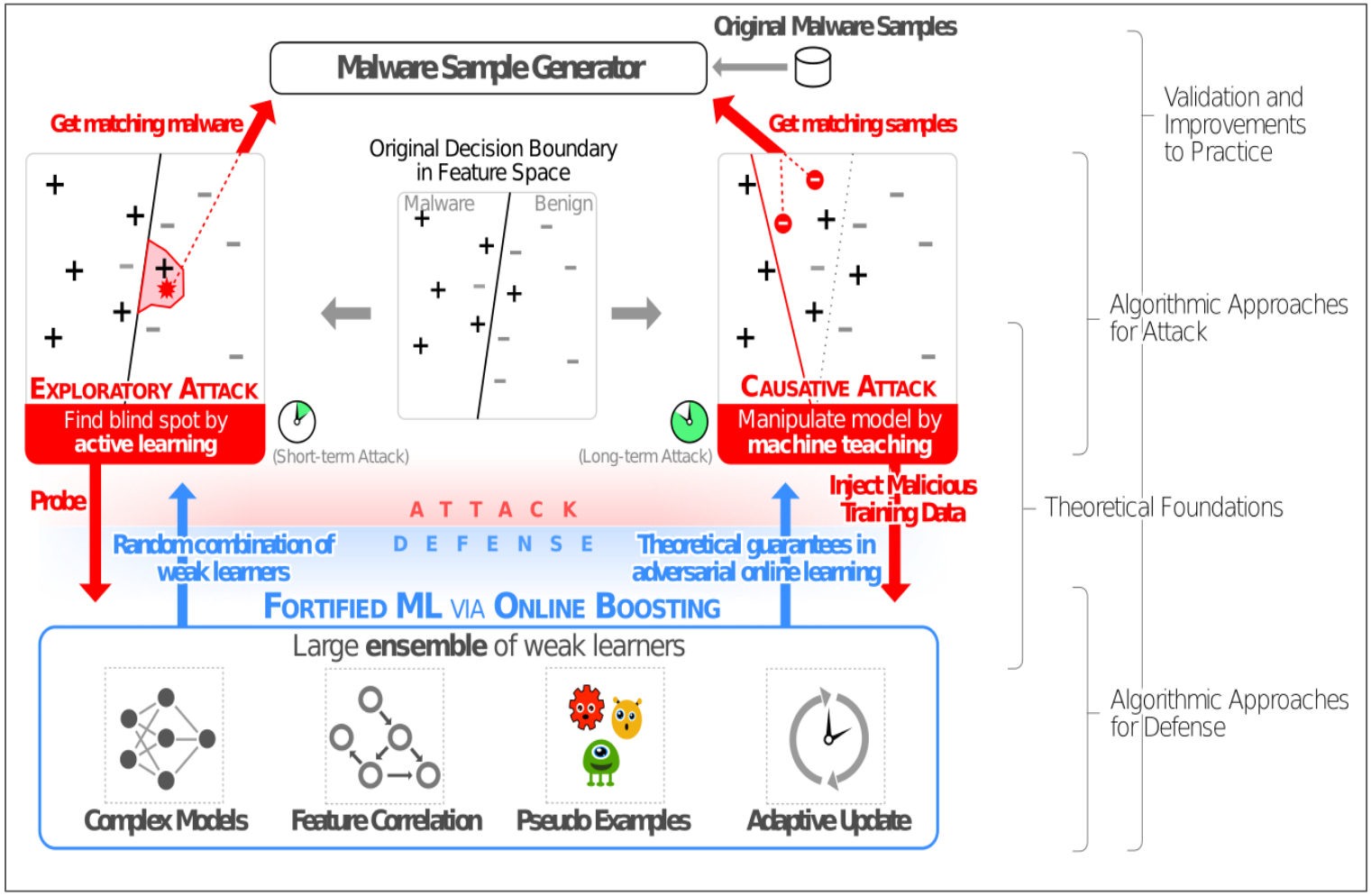


Liveness Detection

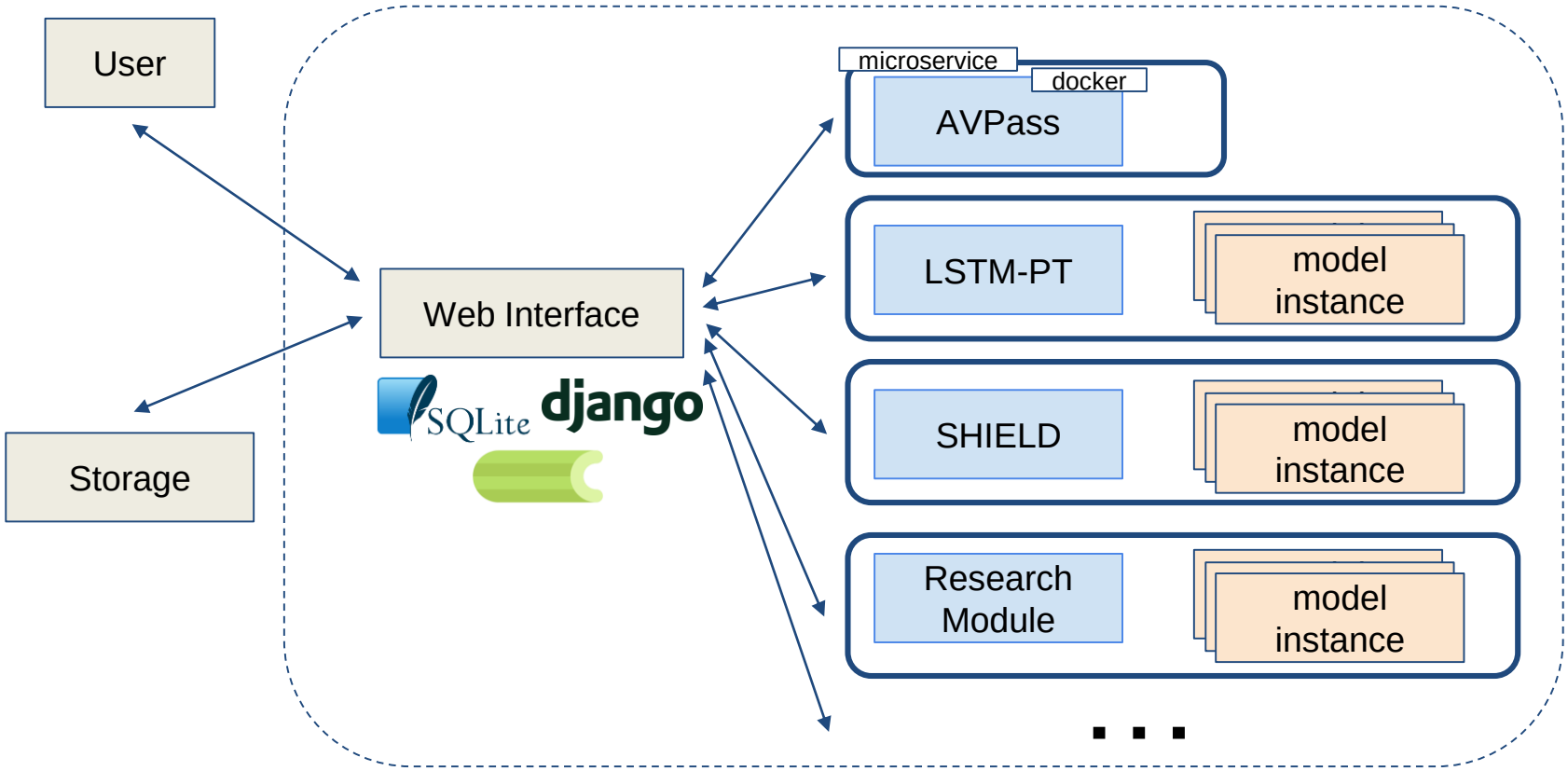
- **Live-biometrics**
  - Enrollment & recovery
  - Ask users to read out random and funny English phrases in CAPTCHA format (rtCapctcha)
    - **Answer** question being asked
    - **Random** to prevent replay attacks



ISTC-Adversarial Resistant Security Analytics



MLsploit


User
Storage
Web Interface
SQLite
django
microservice
docker
AVPass
LSTM-PT
model instance
SHIELD
model instance
Research Module
model instance
. . .

Credits

- Simon Chung
- Brendan Dolan-Gavitt (NYU Poly)
- Yeongjin Jang (Oregon State)
- Bryan Payne (Netflix)
- Chengyu Song (UC Riverside)
- Erkam Uzun
- Tielei Wang (now a start-up in China)
- Carter Yagemann