Gyrus: A Framework for User-Intent Monitoring of Text-Based Networked Applications

Yeongjin Jang*, Simon P. Chung*,
Bryan D. Payne†, and Wenke Lee*
*Georgia Institute of Technology
† Nebula, Inc

Traditional Host-Based Security

- Misuse detection: cannot handle unknown attacks
- Anomaly detection: mimicry attacks

Motivation

- Defining attack is hard
  - 0-day, mimicry attack, and etc…
  - Attacks are keep evolving…
- Then, can we design a security monitor that works for the new attacks?

A New Approach

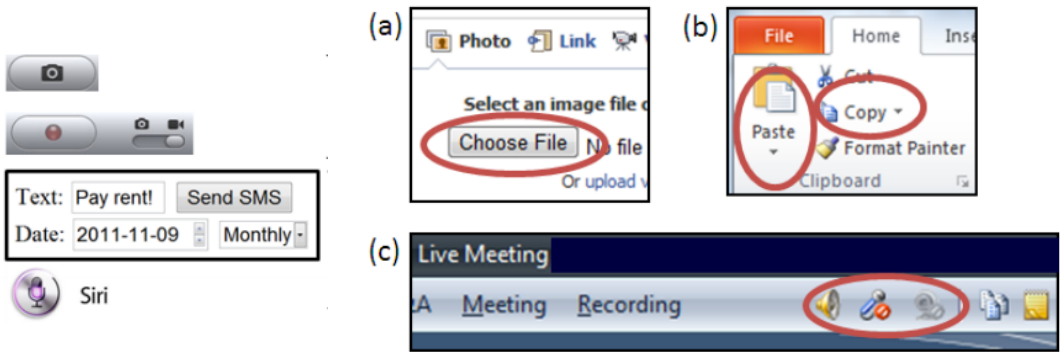
- Objective
  - Protecting *integrity* of user intended text content that will be sent as network packets.
- Attack-agnostic Defense
  - It does not depend on the how the attack works.
    - Examples of the ways of attacks
      - Attach to a process to change some text values…
      - Directly write on /dev/mem to modify sensitive values…
  - We only make sure the monitored system is behaving **correctly**
    - Essentially looking at the opposite side of attack detection.

Related Works

- Using Timing Information
  - BINDER [ACSAC 05', Cui et. al.]
  - Not-A-Bot, [NSDI 09', Gummadi et. al.]
  - User-intent Detection
    - Monitors physical keystrokes/mouse clicks
    - A traffic without user input preceded in a short time window is not user-intended, a malicious activity.
      - User-intended behavior: $T_{network} - T_{input} < T_{threshold}$
  - Simple, but effective defense for existing attacks

Related Works (Cont'd)

- User-Driven Access Control [Oakland 12', Roesner et. al.]
  - Access Control Gadget (ACG)
    - A UI gadget that grants permission to the resource when it is clicked.
      - Examples
        - » Camera icon -> grant access to camera
        - » File-saving icon -> grant access to filesystem



Related Works (Cont'd)

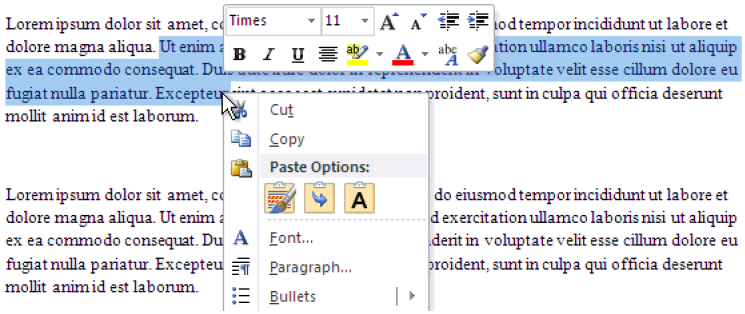
- Problem
  - Only checks existence of user intent (yes/no)
  - BINDER & Not-A-Bot
    - Send malicious network traffic shortly after every keystrokes
  - ACG
    - Free to use the resource after getting of the access
  - Nobody took account into monitoring user-intended content.
    - Why?

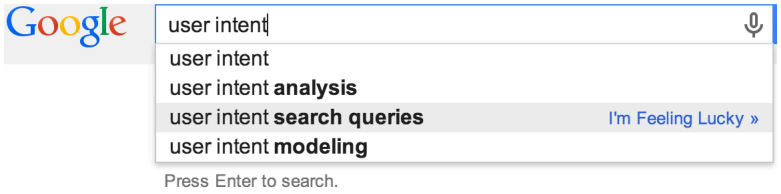
Capturing User-intended Text

- Straightforward way
  - Looking at keystrokes
    - Keycode can be caught at keyboard driver
      - ‘w’, ‘r’, ‘i’, ‘t’, ‘e’, ‘ENTER’
    - Cursor point and button can be caught at mouse driver
      - (x, y, button) -> (325, 641, LCLICK)

Capturing User-intended Text

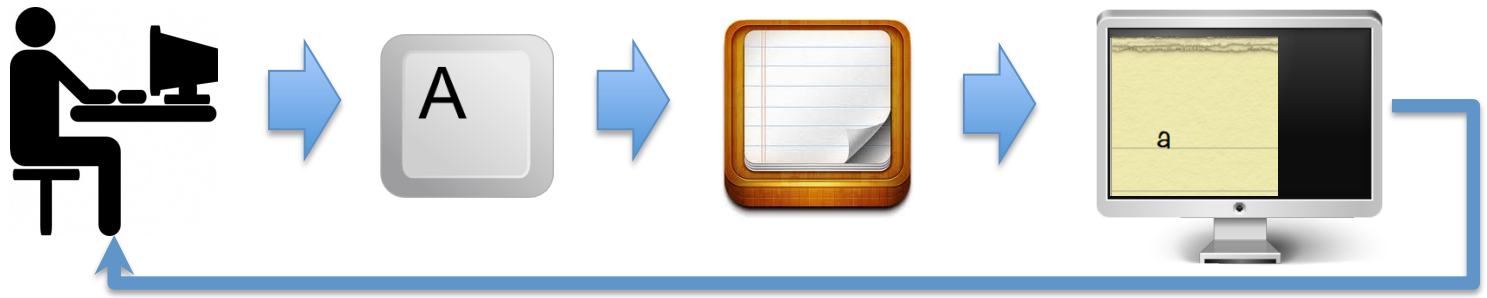
- Challenges
  - Mouse
    - Move cursor on click!
    - Drag to select text, then delete
  - Keyboard
    - Copy & Paste
    - AutoComplete
  - Rich semantics of UI is needed.





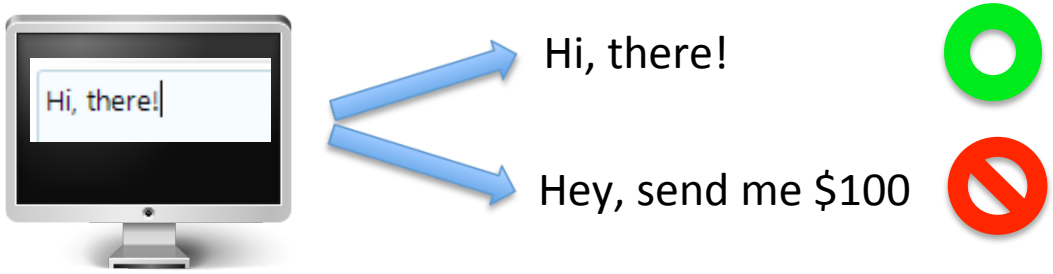
Capturing User-intended Text

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



Capturing User-intended Text

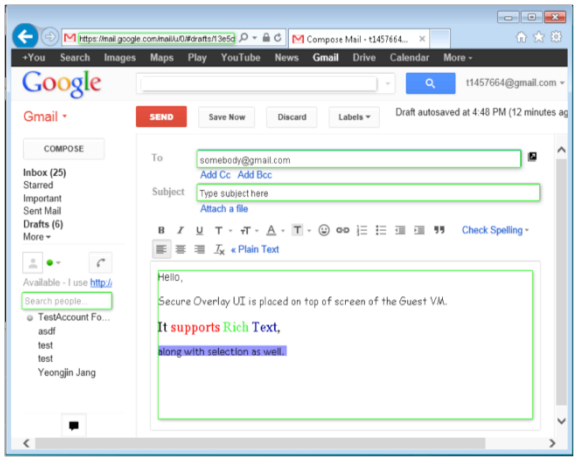
- Observation
  - User naturally verifies what they type by what they sees on the screen
- A New Security Policy
  - What You See Is What You Send (WYSIWYS)
    - We assume on-screen text is user-intended
    - Only allows outgoing traffic that matches on-screen text



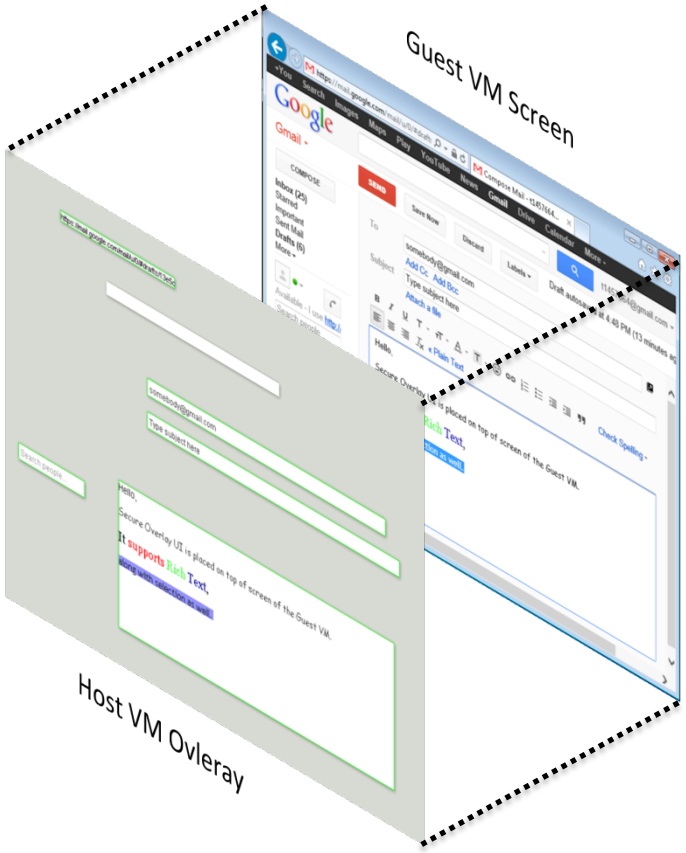
What You See Is What You Send

- WYSIWYS

Secure Overlay



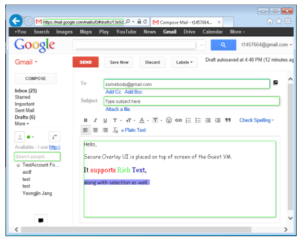
Combined Screen



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

Secure Overlay

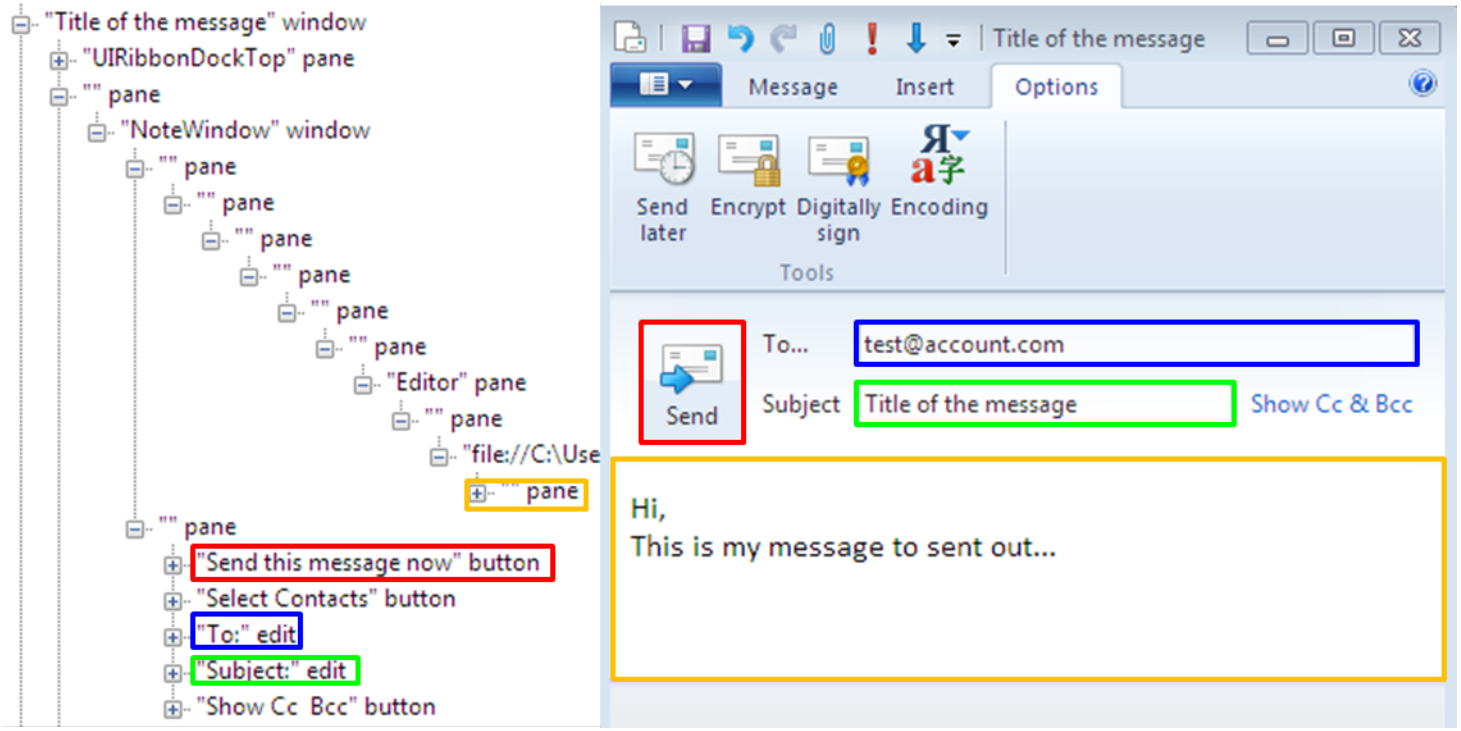
- Only re-draws editbox
  - Exactly same location, size, and color
  - Can support rich-text
    - Font, size, color, style, and etc.
- Passive UI
  - It does not gets any user input.
  - Content will be updated after each applications gets input.
  - Support selection, copy/paste, spell correction, auto-completion, etc…



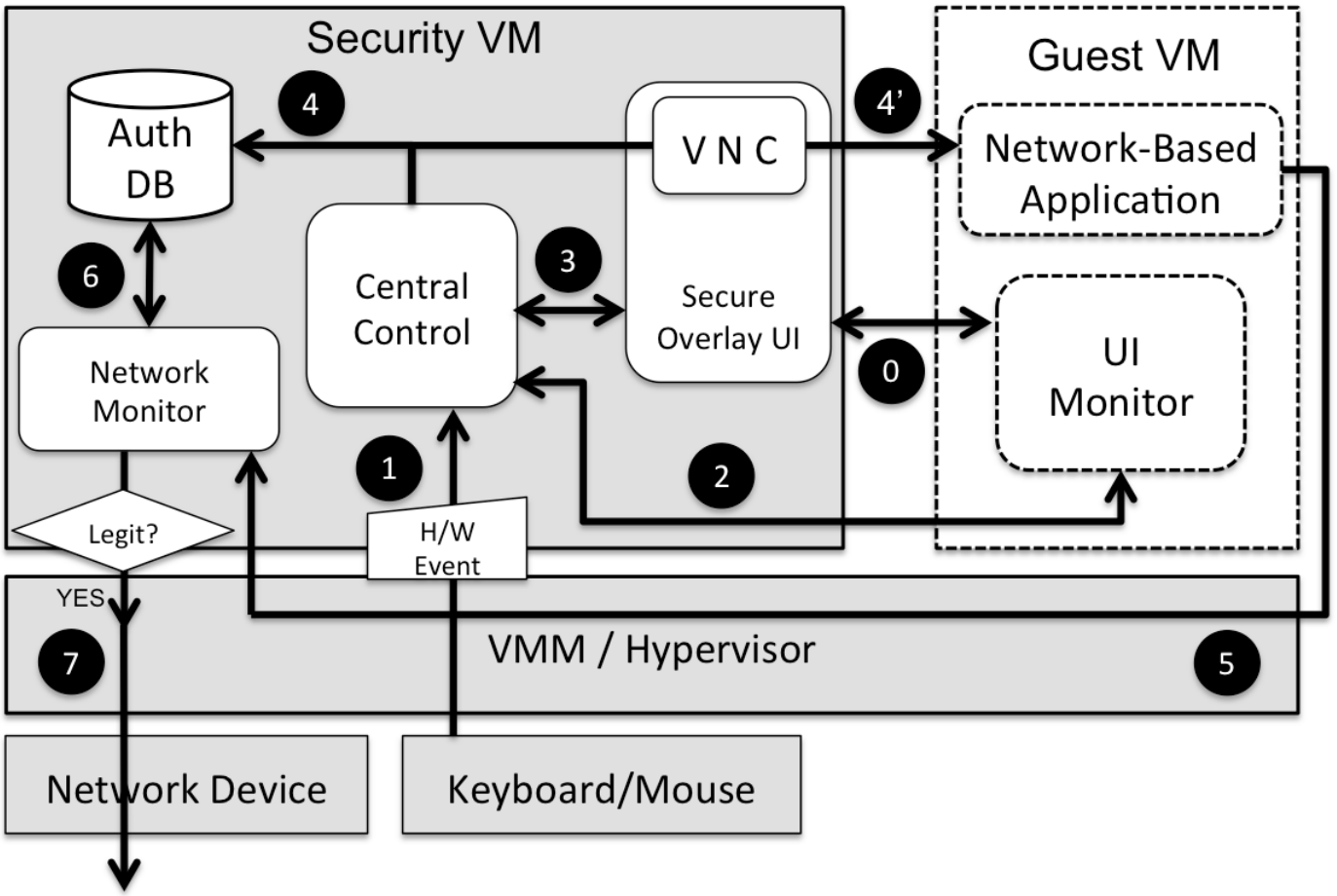
Combined Screen

UI Monitor

- Uses library for UI Testing (UIAutomation)



The Gyrus Architecture



Threat Model

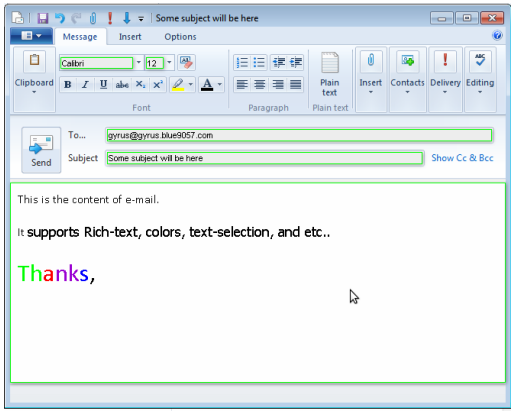
- Hypervisor and security VM is fully trusted.
  - Assumes VM escape is impossible.
- Hardware input devices are trusted, and the attacker has no physical access to it.
  - Attacker cannot forge hardware input event

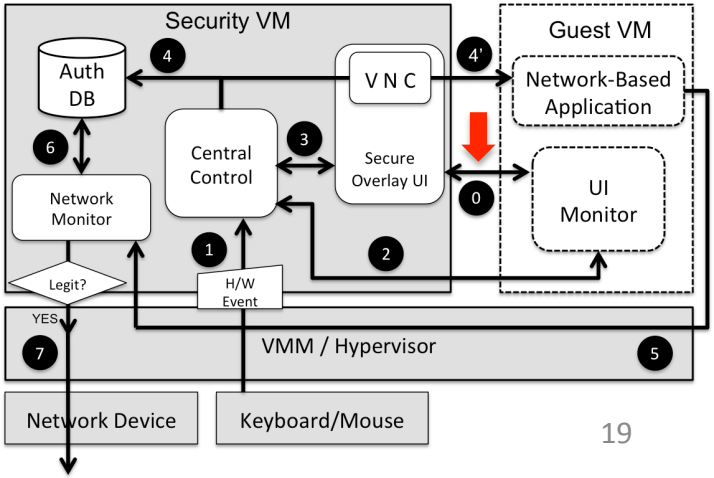
Threat Model (Cont'd)

- 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.
- We completely distrust User VM
  - We allows all attacks including Kernel-level malware.
    - UI monitor is untrusted.

How Gyrus Works

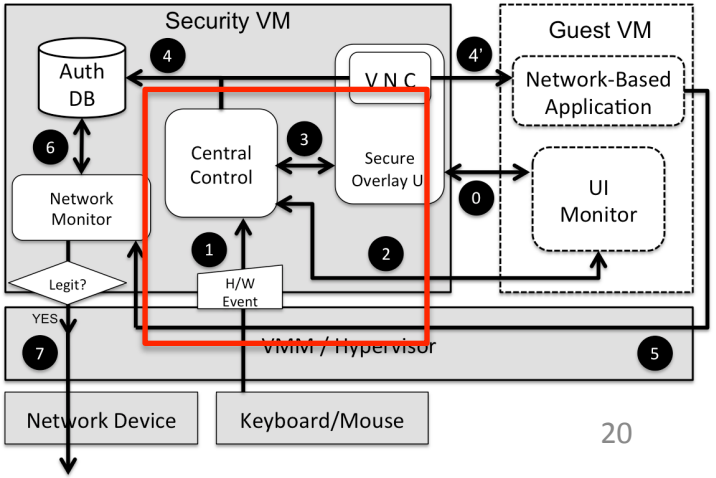
- Identifying and overlaying all editboxes
  - Only shows for focused window
  - Suppress background update
- Track updates
  - Updates all editbox on
    - Change of focus
    - Change of location
    - Change of content





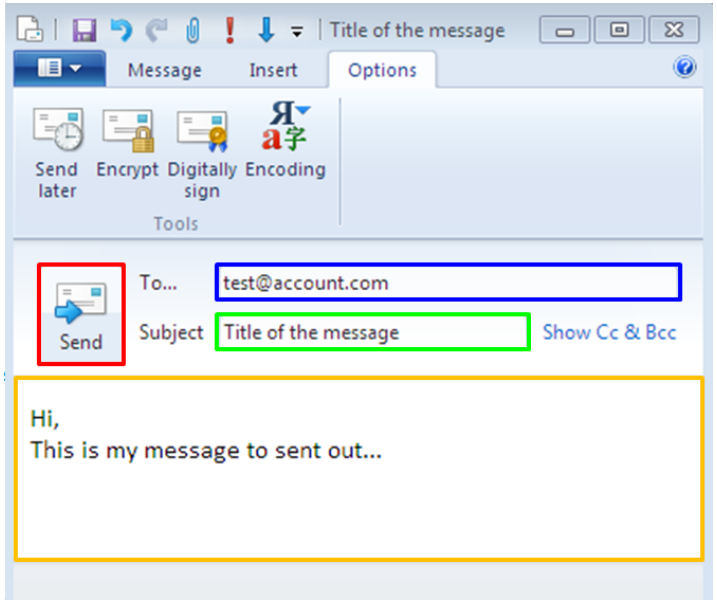
How Gyrus Works

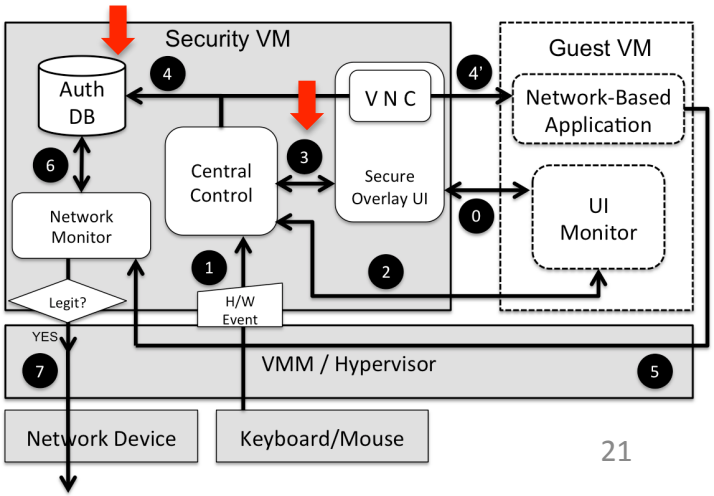
- On every user interaction, checks whether it triggers traffic
  - Traffic-triggering event
    - Click `Send' button on GMail
    - Pressing `ENTER' on facebook message dialog
    - Pressing Ctrl-S on Outlook Express…



Capture User-Intent

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



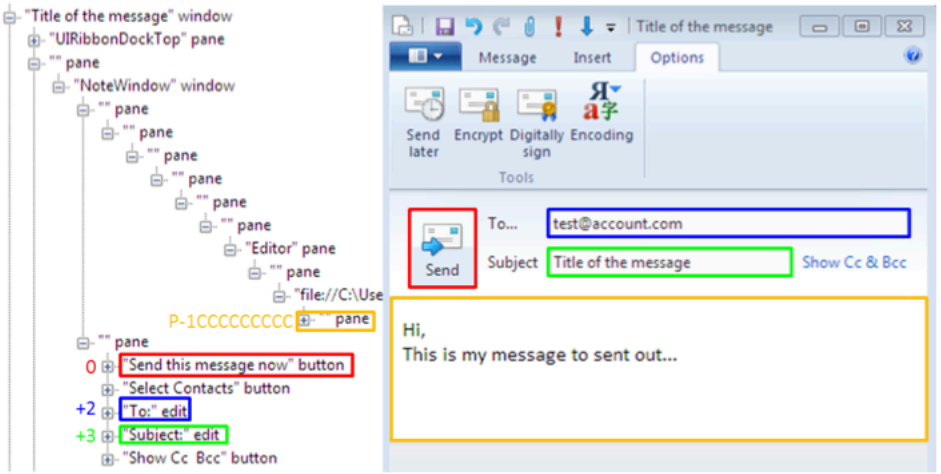


Application-specific Logics

- User Intent Signature

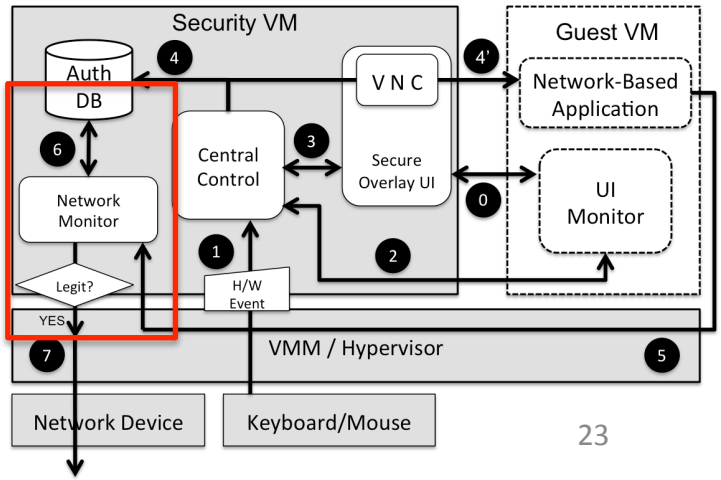
Example 1 User Intent Signature for sending e-mail on Windows Live Mail.

```
{
  "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"
      }
    }
}
```

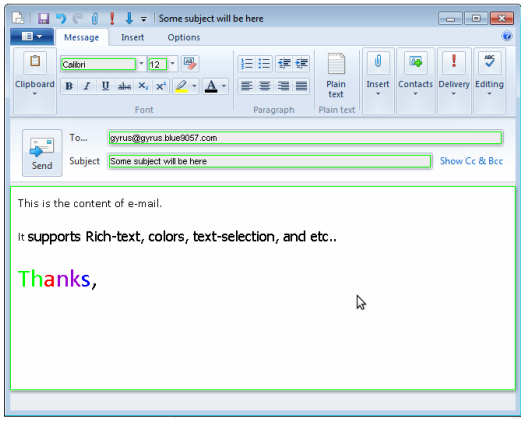


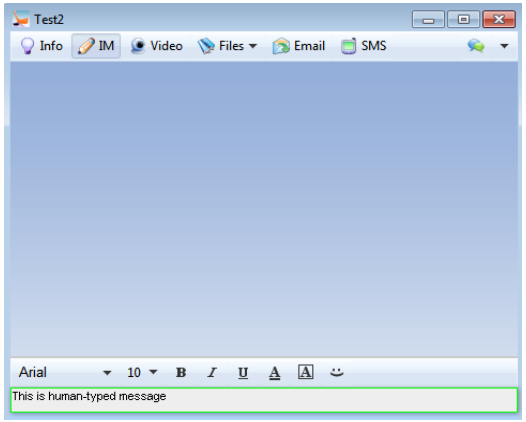
Network Monitor

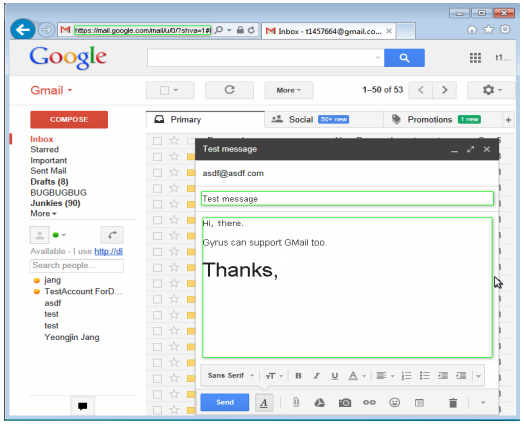
- A transparent proxy with deep-packet inspection
  - Extract user-intent from the traffic, query authorization DB
    - Pass only when it is matched with stored intent…
- Requires proxy per each protocol
- SSL traffic should be decrypted (MITM)

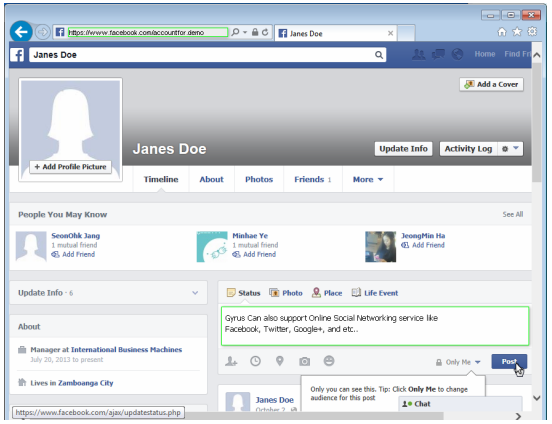


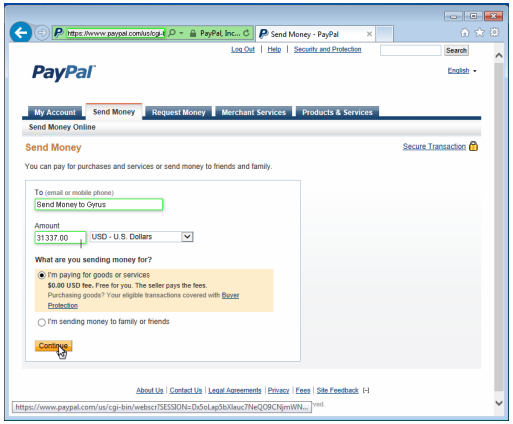
Application Examples











Georgia Institute of Technology

Evaluation

- Security
  - For existing attacks on Apps
    - WYSIWYS is enforced
      - All malware failed to send their traffic on
        - » E-mail client (send spam)
        - » Internet Messenger (send spam)
        - » Facebook (post article, message, and etc.)
        - » Paypal (XSS)
        - » Etc..

Evaluation

- Security
  - Incorrect User Intent Signature
  - On attacking UI monitor in Guest VM
    - Failure on getting correct information
      - False positive, user traffic will be blocked
      - DoS

Evaluation

- Performance
  - Interaction delay
    - Checked turn-around time starting from the input, end with the resulting text or actions on the Overlay
    - Can handle around 1,400 inputs / min (43ms delay)

Actions	Average	STDV	Median	Max
Typing	39ms	21ms	34ms	128ms
ENTER	19ms	6ms	17ms	43ms
LCLICK	43ms	15ms	41ms	79ms
Focus Change	21ms	19ms	17ms	158ms
Move & Resize	21ms	16ms	16ms	85ms

TABLE II. LATENCY INTRODUCED BY GYRUS WHILE PROCESSING THE INPUT. USER-INTERACTION DATA WAS COLLECTED DURING THE USE CASE EVALUATION.

Evaluation

- Performance
  - Network delay

Cases	KVM	Gyrus	Overhead
Single (A)	101.7ms	102.3ms	+0.6ms (.5%)
Single (B)	31.20ms	32.30ms	+1.1ms (3.5%)
Web Page	897.5ms	951.3ms	+53.8ms (6%)
Download	51.1MB/s	49.3MB/s	–1.8MB/s (3.5%)

TABLE III. NETWORK LATENCY FOR HTTP CONNECTION.

Cases	KVM	Gyrus	Overhead
Single Request	90.72ms	94.50ms	+3.78ms (4%)
Download	37.40MB/s	35.23MB/s	–2.17MB/s (5.8%)

TABLE IV. NETWORK LATENCY FOR HTTPS CONNECTION (WITH MAN-IN-THE-MIDDLE PROXY).

Limitations

- Can only handle text so far…..
  - File/Image attachments
    - What we see: name of path (e.g., c:\boot.ini)
    - What machine sends: content of the file
    - Using ACG would be helpful
- Only works if what you see is really what you send
  - Not the case if displayed text undergone a lot of (proprietary) processing before being sent out.
  - However, base64, SSL, and REST API through HTTPS can be handled.

Conclusion

- Gyrus
  - A correct-behavior based monitoring system.
  - Monitors user-intended text through on-screen UI data, and enforcing WYSIWYS policy.
  - Protect most of text-based user applications with minimal overhead.
  - Its attack-agnostic defense works for preventing future attacks.

Questions?

- Q&A