Mimesis Aegis: A Mimicry Privacy Shield

A System’s Approach to Data Privacy on Public Cloud

Billy Lau
Simon Chung
Chengyu Song
Yeongjin Jang
Wenke Lee
Alexandra Boldyreva
INTRODUCTION
Unsatisfactory Status Quo

• Users do not have control over their data that is communicated over public cloud
  – Rely on server to secure user’s data
  – Conflict of interest for data privacy between users and public cloud service (PCS) providers
  – E.g. WhatsApp, Viber, WeChat, etc.
Changing the Status Quo

• Can be solved if users use end-to-end encryption
  – Hard to use in practice

• Existing solutions:
  – Requires user to be trained to use custom apps to perform safe communication
  – Have questionable data isolation model
Mimesis Aegis

• Applies end-to-end encryption to users’ communication data while preserving user experience by:
  – Mimicking GUIs of app-of-interest
  – Interacting with app-of-interest on behalf of user

• Good isolation model

• Generalizable across different apps in the same category

• Resilient to app updates
Mimesis Aegis - WhatsApp
Mimesis Aegis - WhatsApp

Hi
6:13 AM

Hi, how you doing?
6:13 AM

Great. The meeting is on 5pm, right?
6:14 AM

Yes. Don’t forget to bring the documents.
6:14 AM

Alright. See you then.
6:15 AM

BTW, where is the place to meet?
6:15 AM

at bldg 3.
RELATED WORK
Standalone Solutions

• Protect data confidentiality
• Good isolation from untrusted entities
• Examples: PGP, Gibberbot, TextSecure, SafeSlinger, FlyByNight, etc.

• Problem:
  – Requires open protocol
  – Do not preserve user experience
Browser Plugins/Extensions

• Provides transparent integration with applications of interest

• Examples:
  – Scramble!, TrustSplit, NOYB, SafeButton, etc.

• Problem: Only applicable to web applications.
  – How about mobile devices?
App Rewriting/Repackaging

• Provides transparent integration with applications of interest

• Examples:
  – Aurasium, Dr. Android, etc.

• Problems:
  – Breaks app updates
  – The security of the reference monitor may be compromised as it resides in the same address space as the untrusted entity
SYSTEM DESIGN
Design Goals

• Offer good security
  – Strong isolation from untrusted entities

• Preserve user experience
  – Transparent interaction with existing apps

• Easy to maintain and scale
  – A sufficiently general-purpose approach
Threat Model

• Untrusted parties:
  – Public cloud service (PCS) providers
  – Client-side apps
  – Middle boxes between a PCS and client-side app
Threat Model

• Trusted components:
  – Hardware
  – Operating System (OS)
  – Soft keyboard
  – M-Aegis components
  – The user
M-Aegis Architecture

• Layer 7.5
M-Aegis Architecture

• UI Automation Manager (UIAM)
  – Gives M-Aegis the context of the screen
  – Provides information to correctly render mimic GUIs on L-7.5
  – Relays user input to the underlying app
M-Aegis Architecture

• Per-Target Client App (TCA) Logic
  – Processes UI tree to determine a TCA’s current UI state
  – Makes sense out of the information gathered from UIAM
  – Decides suitable actions for different UI states
M-Aegis Architecture

• Cryptographic Module
  – Key Manager
  – Searchable Encryption Scheme
    • Easily-Deployable Efficiently-Searchable Symmetric Encryption Scheme (EDESE)
Easily-Deployable Efficiently-Searchable Symmetric Encryption Scheme (EDESE)

• Main idea – tag the encrypted text
  – Bad idea: leaving ciphertext of keywords in the open.

• Utilize bloom filter (BF) to “collect” keywords.
  – Problem: email providers don’t support BF tests.
  – Solution: cleverly encode BF in such a way that it is searchable by simple string matching.
Easily-Deployable Efficiently-Searchable Symmetric Encryption Scheme (EDESE)

- **Algorithm:**
  - Determine the size of BF, \( n \) bits.
  - Determine the \# of bits to be turned on in the BF, \( k \) bits.
  - For every unique keyword in the text:
    - Produce \( k \) keyed hash digest, \( m_i \).
    - Treat every \( m \) as a the position in the BF to be turned on.
  - Encode all the positions that are ON into text.
  - Append the encoded BF to the encrypted text.
USER WORKFLOW
Demo Video with WhatsApp
Performance Evaluations

• Experimental Setup:
  – Stock Android phone (LG Nexus 4)
    • Android 4.4.2 (Kit Kat, API Level 19)
  – Each experiment is repeated 10 times and the average is taken
Performance Evaluations

• Preview Encrypted Email:
  – 76 ms to render plaintext on L-7.5
  – Well within expected response time (50 – 150 ms)

• Composing and Sending Encrypted Email:
  – Used Enron Email Dataset
  – With longest email:
    • 953 words, of which 362 are unique
    • 205 ms to encrypt, build the search index, and encode
Discussions

• Limitations:
  – Not robustly tested against social engineering attacks
  – Currently only handles text-based apps
  – Does not tolerate typographical error during search
Conclusions

• Users can now regain control over their private data using Mimesis Aegis, where:
  – Plaintext is never visible to client apps
  – Original user experience is preserved
  – Technique is generalizable to a large number of apps and is resilient to app updates
Questions?