Privacy Enhancing Technologies (PETs) Testbed

Partha Das Chowdhury (partha.daschowdhury@bristol.ac.uk)
Joe Gardiner (joe.gardiner@bristol.ac.uk)
Testbed Team

Awais Rashid
Tariq Elahi
Joe Gardiner
Partha Das Chowdhury
Mohammad Tahaei

Student Developers:
Graham Peden (Networking)
Maysara Alhindi (Orchestration)
Winston Ellis (Orchestration)
Maria Sameen (Modelling)
Anthony Mazeli (Documentation)
Jacob Halsey (Project Student)
Overview

• Background
• Use Cases
• Design Considerations
• Current Implementation
• Demonstration
• Privacy Analysis
• Future Work
• Conclusion
Use Cases
Diversity of Users

Research
- CS, HCI, STS, Funders,
  - Publication, Evaluation, Teaching

Policy
- Standardization, Law Enforcement, Civil Servants
  - Regulatory Content/Enforcement, Sharing datasets

Public Users
- Journalists, Special Abilities, Charities,
  - Citizen Watch

Industry
- Developers, Managers
  - Regulatory Evaluation, Independent Evaluation, Use Datasets
Use Case 1

- Developer Alpha produces an app using multiple third party libraries
- Wants to see if libraries are collecting unnecessary data from users
- Testbed launches multiple instances of Android and iOS devices with app installed
  - Testbed can simulate user interaction with app
- Testbed collects all network traffic from apps to internet, presents report to Alpha
  - Traffic contents, destinations etc
- Testbed can map collected data to a privacy-evaluation framework (e.g. Privacy by Design, LINDDUN)
- Testbed can apply automated analysis (e.g. Exodus, LibRadar)
Use Case 2

• Developer Beta develops a privacy preserving P2P file sharing application
• Wants to measure resilience against attacks such as Sybil or partitioning
• Launches large number of instances in P2P topology
• Makes subset of instances “malicious” to perform the attack
• Performs attacks, and measures impact on privacy and performance
Use Case 3

- Privacy Engineer Gamma wants to learn about and test modern PETs, e.g. homomorphic encryption, secure multi-party computation and differential privacy
- Testbed used to run and evaluate these technologies before use in final product
  - Can launch instances and simulate “users”
Current Test Case – End to End Encrypted Messengers

• What information is leaked from end to end encrypted messengers?
  • E.g. Signal, Whatsapp, Telegram

• Run clients in testbed, capture traffic, look for information leakage
  • Metadata
  • Inferable data
Testbed Design Considerations
Key Functionalities

• Deployment
• Orchestration
• Data Logging
Deployment

• Testbed should allow for easy deployment of services and hosts
  • Potentially thousands
• Support for both traditional hosts, as well as emulated smartphone OSs
• Testbed should provide a virtual network
  • Use of SDN for orchestration
Orchestration

• Testbed should allow for automated control of applications
• Simulated user interaction, simulated sensor values
• Replaying of network traffic captures
Data Logging

• Testbed should capture sufficient data for analysis

• Potential sources:
  • Network captures
  • Memory captures
  • Screen captures
Further Design Elements

• Application Agnostic
  • Testbed should support multiple application types and architectures

• Extensibility
  • Testbed should be scalable.
  • Multiple instances of testbed should be joinable to increase virtualisation capability

• Automated Analysis
  • Testbed should have automated privacy analysis tools to be easily applied to use cases with minimal knowledge

• Modularity
  • New features (such as new analysis tool) can be added to testbed with ease
Testbed Implementation
• Testbed consists of tool, kvm-compose, which manages deployment, networking and orchestration.
Virtualisation

• Virtualisation is provided using KVM
• Can deploy OS from disk image, or build as required
• Android applications emulated using Google’s Android Virtual Device (AVD)
  • Deployed inside Ubuntu Desktop VM
• Virtualisation managed by kvm-compose tool
kvm-compose

• kvm-compose is a CLI tool Jacob developed (and expended by team) for Linux using Rust (and the libvirt library) that takes in a custom configuration file format describing a test environment, and can create or destroy it (with up/down subcommands).

$ kvm-compose up

• This is the first step of automating a testbed:
  • From a simple configuration file kvm-compose will deal with the conversion to a relatively complex libvirt domain configuration XMLs (for KVM), and create the virtual machines.
  • It will also create and connect the virtual machines up to a virtual network
Cloud-init, Scripting, Context etc.

- **cloud-init** is used to automatically initialize new virtual machines (disk creation and software installation). The NoCloud datasource option uses a clever system of attaching a specifically formatted virtual disk, and passing flags via the SMBIOS serial number of the VM.

- What happens now?
  - The Ubuntu cloud image will be downloaded (once and then cached), copied, and expanded with 20G free space.
  - At boot the machine will auto configure its hostname to match the machine name in the configuration file.
  - Our SSH public key will be injected into the instance, allowing remote access.
  - Files in the context folder will be copied in at /etc/nocloud/context.
  - An arbitrary run_script will be run once on the first boot.
Networking – Software Defined Networking

• Networking is provided using OpenvSwitch (OVS)

• OVS bridges can easily be linked up to an SDN controller (such as Floodlight), enabling more advanced network management.
Networking - Management

- Linux bridge acts as DHCP and DNS server for clients, and as gateway for internet access
- Also provides NAT service for external connections
- At least one of the OVS bridges must be connected to Linux bridge
- Clients connect to OVS bridges, Linux bridge assigns IP addresses and runs internal DNS
- Internal routing managed by SDN controller, external traffic routed to Linux bridge and external interface
Orchestration

• Need to be able to configure and control applications automatically
• Cloudinit can handle configuration and installation
• For interaction with command line applications, can use SSH
• For automated interaction with AVD devices, can use ADB functionality
  • Send screen presses
  • Send text
  • Can be recorded and replayed, or programmatically generated
Data Capture

• Tcpdump can be run on OVS bridges
  • A mirror port is configured and used for capture
• Can capture on individual bridges to capture at different points on the network
• If using external services, can capture on Linux Bridge to collect all traffic from testbed to outside
  • Does feature noise from background host machine and VM traffic
Demonstration
Demonstration Overview

• We will launch an deployment of the Signal end-to-end encrypted messaging application
• We will run 2 Android Signal clients on emulators, communicating with the real-world Signal servers
  • We can also run the 3rd party Signal-CLI client with our own server
• Clients will register numbers manually
• Clients will communicate automatically
• Testbed is running on a Dell Workstation
  • 2x Intel Xeon 4110 CPU (8 core 2.1 GHz), 125Gb RAM.
Demonstration – Virtual Machines

• We launch two instances of an Ubuntu VM running ADB to create 2 Signal Clients

machines:
- name: client1
  cpus: 2
  memory_mb: 6144
  extended_graphics_support: true
  disk:
    existing_disk:
      path: /home/spedam/prj/rephrain-testbed/demos/ubuntu20.04-1.qcow2
      driver_type: qcow2
      device_type: disk
      readonly: false
  interfaces:
    - bridge: br0
      run_script: ./run.sh
      context: /emulator/

- name: client2
  cpus: 2
  memory_mb: 6144
  extended_graphics_support: true
  disk:
    existing_disk:
      path: /home/spedam/prj/rephrain-testbed/demos/ubuntu20.04-1.qcow2
      driver_type: qcow2
      device_type: disk
      readonly: false
  interfaces:
    - bridge: br0
      run_script: ./run.sh
      context: /emulator/

bridges:
- name: br0
  controller: tcp:127.0.0.1:6653
  protocol: OpenFlow13

external_bridge: br0
sshd_public_key: ___
password_ssh_enabled: true
Compose Up
Demonstration - Network

[Diagram of network components including Internet, Host Ethernet - Fixed IP, LinuxBridge (DHCP, DNS via libVirt), ovs Br0, Floodlight, Client1, Client2, and Testbed]

Signal Server (AWS)
Floodlight SDN
Signal Registration
Orchestration

• Clients automatically send messages to each other
  • Includes simulated picture messages controlled by finger clicks

• Launched by bash script, messages sent using ADB functions
  • Automated text entry, no user interaction required

```bash
#!/bin/bash

send_image()
{
    echo "Sending an image"
    adb exec-out input tap 195 379
    sleep 1
    adb exec-out input tap 161 495
    sleep 1
    adb exec-out input tap 291 597
    sleep 1
}

send_text()
{
    echo "Sending text $1"
    adb exec-out input text "$1"
    sleep 1
    adb exec-out input tap 284 380
    sleep 1
}

click_on_keyboard()
{
    adb exec-out input tap 126 611
}

while :
do
    click_on_keyboard
    send_text "hello"
    send_image

done
```
Client Messaging (Automated)
Demo – Data Capture

- Traffic is captured on OVS bridge
- Saved to PCAP
  - One capture during registration
  - One during messaging
- Used for privacy analysis
Privacy Analysis
Overview

• The information a passive adversary observing the network can learn.
  • Can they link communicating entities?
  • The hosts that has access to network information and information they can reveal.

• The information an active adversary can get –
  • For example, deanonymizing the sender and the recipient

• Information captured by third party APIs

• The efficacy of shareable datasets
  • Compliance with adequate anonymization

• Systematically analyse privacy behaviour of client-server application
  • Integrate with privacy threat elicitation frameworks the information visible to passive and active adversary
  • Evolve a risk/threat scenario
Extracted features

- Source IP Address
- Destination IP Address
- Source Port
- Destination Port
- Domain Name
- IP Protocol Specifier
- Length of packet
- Traffic Class - QoS
How we performed the analysis?

Step 1: Packet Capture
Step 2: Initially features available
Step 3: Search for possible features
Step 4: Extraction of more features
Step 5: Identification of significant features
Step 6: LINDDUN
Sample Results – Signal Message Exchange

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Domain Name</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.248.212.111</td>
<td><code>ac88393aca5853df7.awsglobalaccelerator.com</code></td>
<td>United States</td>
</tr>
<tr>
<td>142.250.178.10</td>
<td><code>lhr48s27-in-f10.1e100.net</code></td>
<td>United States</td>
</tr>
<tr>
<td>142.250.180.4</td>
<td><code>lhr25s32-in-f4.1e100.net</code></td>
<td>United States</td>
</tr>
<tr>
<td>142.250.200.10</td>
<td><code>lhr48s29-in-f10.1e100.net</code></td>
<td>United States</td>
</tr>
<tr>
<td>142.250.200.35</td>
<td><code>lhr48s30-in-f3.1e100.net</code></td>
<td>United States</td>
</tr>
<tr>
<td>142.250.200.42</td>
<td><code>lhr48s30-in-f10.1e100.net</code></td>
<td>United States</td>
</tr>
<tr>
<td>192.168.222.1</td>
<td><code>ip-192-168-222-1.eu-west-2.compute.internal</code></td>
<td>Not found</td>
</tr>
<tr>
<td>192.168.222.217</td>
<td><code>ip-192-168-222-217.eu-west-2.compute.internal</code></td>
<td>Not found</td>
</tr>
<tr>
<td>192.168.222.7</td>
<td><code>ip-192-168-222-7.eu-west-2.compute.internal</code></td>
<td>Not found</td>
</tr>
<tr>
<td>216.58.212.202</td>
<td><code>ams16s21-in-f10.1e100.net</code></td>
<td>United States</td>
</tr>
<tr>
<td>216.58.212.206</td>
<td><code>ams16s21-in-f14.1e100.net</code></td>
<td>United States</td>
</tr>
<tr>
<td>35.232.111.17</td>
<td><code>17.111.232.35.bc.googleusercontent.com</code></td>
<td>United States</td>
</tr>
<tr>
<td>76.223.92.165</td>
<td><code>ac88393aca5853df7.awsglobalaccelerator.com</code></td>
<td>United States</td>
</tr>
</tbody>
</table>

DST-IPs, Name & Location

Location on Map
Signal Client Registration

13.248.212.111, 'ac88393aca5853df7.awsglobalaccelerator.com', United States
142.250.178.19, 'lhr48s27-in-f19.1e100.net', United States
142.250.178.2, 'lhr48s27-in-f2.1e100.net', United States
142.250.179.227, 'lhr25s31-in-f3.1e100.net', United States
142.250.179.234, 'lhr25s31-in-f10.1e100.net', United States
142.250.180.10, 'lhr25s32-in-f10.1e100.net', United States
142.250.180.4, 'lhr25s32-in-f4.1e100.net', United States
142.250.200.10, 'lhr48s29-in-f10.1e100.net', United States
142.250.200.35, 'lhr48s30-in-f3.1e100.net', United States
142.250.200.42, 'lhr48s30-in-f10.1e100.net', United States
142.250.200.46, 'lhr48s30-in-f14.1e100.net', United States
142.251.5.188, 'wg-in-f188.1e100.net', United States
172.217.16.234, 'mad08s04-in-f10.1e100.net', United States
172.217.169.10, 'lhr25s26-in-f10.1e100.net', United States
172.217.169.74, 'lhr48s09-in-f10.1e100.net', United States
192.168.222.1, 'ip-192-168-222-1.eu-west-2.compute.internal', Not found
216.58.212.206, 'ams16s21-in-f206.1e100.net', United States
216.58.213.10, 'lhr25s25-in-f10.1e100.net', United States
34.122.121.32, '32.121.122.34.bc.googleusercontent.com', United States
35.232.111.17, '17.111.232.35.bc.googleusercontent.com', United States
76.223.92.165, 'ac88393aca5853df7.awsglobalaccelerator.com', United States
What is LINDDUN?

Model the system
• create a data flow diagram
• describe all data

Elicit threats/risks
• map threats to DFD elements
• identify threats using threat trees

Manage threats
• prioritize in dialog with the DPO
• mitigate using a taxonomy of PETs

Knowledge support
• mapping table
• LINDDUN threat taxonomy

Knowledge support
• Taxonomy of mitigation strategies
• Classification of privacy solutions

Reference: https://www.linddun.org/linddun
LINDDUN Threat Categories

**Linkability**
An adversary is able to link two items of interest without knowing the identity of the data subject(s) involved.

**Identifiability**
An adversary is able to identify a data subject from a set of data subjects through an item of interest.

**Non-repudiation**
The data subject is unable to deny a claim (e.g., having performed an action, or sent a request).

**Detectability**
An adversary is able to distinguish whether an item of interest about a data subject exists or not, regardless of being able to read the contents itself.

**Disclosure of information**
An adversary is able to learn the content of an item of interest about a data subject.

**Unawareness**
The data subject is unaware of the collection, processing, storage, or sharing activities (and corresponding purposes) of the data subject's personal data.

**Non-compliance**
The processing, storage, or handling of personal data is not compliant with legislation, regulation, and/or policy.

Reference: https://www.linddun.org/linddun
Step 6 - LINDDUN

Application Layer

Signal App

Network Layer

Packet (Metadata)

Signal Server
## Step 6 - LINDDUN

<table>
<thead>
<tr>
<th>Metadata</th>
<th>L</th>
<th>I</th>
<th>N</th>
<th>D</th>
<th>D</th>
<th>U</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Src IP Add.</strong></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td><strong>Dst. IP Add.</strong></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td><strong>Src. Port</strong></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td><strong>Dst. Port</strong></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td><strong>Domain Name</strong></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td><strong>IP Protocol Specifier</strong></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td><strong>QoS</strong></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## LINDDUN - Examples and Implications

<table>
<thead>
<tr>
<th>Metadata</th>
<th>L</th>
<th>I</th>
<th>N</th>
<th>D</th>
<th>D</th>
<th>U</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Src IP Add.</strong></td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Dst. IP Add.</strong></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Src. Port</strong></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td><strong>Dst. Port</strong></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td><strong>Domain Name</strong></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td><strong>IP Protocol Specifier</strong></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td><strong>QoS</strong></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Future Work
Future Work – Implementation

- Improved scalability
  - Deployable across multiple machines
- Greater degree of automated interaction
- Further improvements to deployment mechanisms
  - E.g. snapshots
- Implement further data sources for analysis
Future Work – Privacy Analysis

• Pruning the capture information.
• Explore the extent to which we can automate the analysis with LINDDUN
• Incorporate data taxonomy with threat taxonomy.
Publications

• “A Privacy Testbed for IT Professionals: Use Cases and Design Considerations” J. Gardiner, M. Tahaei, J. Halsey, T. Elahi, A Rashid; 7th Workshop on Security Information Workers (WSIW 2021) (Extended Abstract)

Thank You!

Questions?
Contact:
Partha Das Chowdhury (partha.daschowdhury@bristol.ac.uk)
Joe Gardiner (joe.gardiner@bristol.ac.uk)