REPHRAIN

Protecting citizens online

Building a Privacy Testbed: Use Cases and Design Considerations

Joseph Gardiner, University of Bristol

Partha Das Chowdhury, University of Bristol

Jacob Halsey, University of Bristol

Mohammad Tahaei, University of Bristol

Tariq Elahi, University of Edinburgh

Awais Rashid, University of Bristol

October 2021

UK Research and Innovation · University of Bristol · UCL · The University of Edinburgh · King's College London · University of Bath

Building a Privacy Testbed: Use Cases and Design Considerations

Joseph Gardiner[1], Partha Das Chowdhury[1], Jacob Halsey[1], Mohammad Tahaei[1], Tariq Elahi[2], and Awais Rashid[1]

[1] University of Bristol
[2] University of Edinburgh
{joe.gardiner, partha.daschowdhury, vw18148, mohammad.tahaei, awais.rashid}@bristol.ac.uk, t.elahi@ed.ac.uk

Abstract. Mobile application (app) developers are often ill-equipped to understand the privacy implications of their products and services, especially with the common practice of using third-party libraries to provide critical functionality. To add to the complexity, most mobile applications interact with the "cloud"—not only the platform provider's ecosystem (such as Apple or Google) but also with third-party servers (as a consequence of library use). This presents a hazy view of the privacy impact for a particular app. Therefore, we take a significant step to address this challenge and propose a testbed with the ability to systematically evaluate and understand the privacy behavior of client server applications in a network environment across a large number of hosts. We reflect on our experiences of successfully deploying two mass market applications on the initial versions of our proposed testbed. Standardization across cloud implementations and exposed end points of closed source binaries are key for transparent evaluation of privacy features.

Keywords: privacy-enhancing technologies · testbed · usable privacy · privacy professionals.

1 Introduction

For developers privacy is often not the explicit goal [5]. The benefits of making use of fine-grained personal information are immediate, but the consequences of this insecure behavior is delayed and difficult to comprehend [3]. Furthermore, the software collecting and processing personal information encapsulates complex mathematics, tools, and a diverse understanding of privacy. Moreso when Privacy Enhancing Technologies (PETs) are integrated into apps as a mitigation against unwanted data leaks.

Solove et al. argue that privacy is far too complex to be left in the hands of average consumers (including developers); the solution lies in regulating the infrastructure that collects, stores, and transfers information [21]. However, it is currently not possible to gain insights into these infrastructures due to an absence of a mechanism to ascertain the flow of information in practice. This

absence impedes developers, regulators, and users to verify the claims made by applications about their data practices and the PETs they employ.

In this paper, we address this gap by proposing a privacy testbed. We sketch use-cases, discuss design considerations, and reflect on the initial implementations of our proposed **automated testbed** to verify the privacy features/claims of mass market client server applications that use PETs. This forms the basis of the testbed proposed by the National Research Centre on Privacy, Harm Reduction and Adversarial Influence Online (REPHRAIN) announced by UK Research and Innovation in October 2020. While testbeds have been proposed in other settings, e.g., security of control systems and IoT [11, 18, 14], to our knowledge, this paper is the first to propose a privacy testbed.

2 Use Cases

Our proposed testbed can assist software developers, system administrators, and privacy professionals, to run large scale analysis without the need to deploy any infrastructure or have access to several (potentially costly) target devices. They will be able to instantiate multiple virtual devices with various versions of operating systems to facilitate executing privacy-related analyses. Regulators can use our testbed as well for certification and verification purposes. We outline three sample use cases for exposition.

2.1 Contact Tracing Applications

A developer of a contact tracing app uses the Google Apple exposure notification (GAEN) framework [13]. The application uses exposure notification framework to detect individuals who might be exposed to other individuals with a virus. The cryptographic operations are handled by GAEN. The app developers are required to use the *ExposureNotificationClient* class to implement functions allowing users to start/stop tracing, handle exposure related notifications, medical information and receive broadcasts. There is a ephemeral key which is generated at regular intervals and upon infection the history of the keys over a fixed period of time are sent to the authorities for alerting potential contacts within that period. Applications should not reveal any personal sensitive information either during the exchange, broadcasts or while data is at rest. Recent research suggests the security and privacy of contact tracing applications are fraught with imperfections [23].

Our testbed would allow the users to run multiple tests on both the server and the client side using multiple virtual instances. For example, at the client side potential concerns like *can a user de-anonymize infected contacts or other contacts using the app?* can be tested using our testbed. The *enormity and scale of server side data* can be independently explained to regulators, developers and/or end users. Furthermore the data can be interfaced with privacy evaluation frameworks like (Privacy by Design [16] and LINDDUN [9]) to preempt a repeat of CARE data scandals [19]. The ability to refute through practical manifestations of the threat will lead to effective and privacy enhancing application development.

This would be useful in authentication situations (e.g., Kerberos deployments) where privacy is not a requirement but the remote entity is untrustworthy.

2.2 Privacy Preserving Peer to Peer (P2P) File Sharing Systems

Participants in P2P systems also run the infrastructure [24] and rely on the honesty and competence of other participants. One way to disrupt the system is to infiltrate the membership of the network through favored pawns and gain control [4]. Wang et al. describe possible horizontal and vertical attacks to put enough traffic in the hands of the attacker to identify participants in the network [22]. Some solutions suggest the presence of a strong central authority to prevent hostile takeover of the network [10], which leads to a single point of failure. The threat of partitioning by malicious participants generally applies to cryptographic ways of stamping digital documents [15], decentralized property [8], and programmable replicated state machines based upon the Byzantine General's problem [17].

Our testbed can replicate large number of independent instances through virtualization. This gives the ability to deploy a large number varied independent instances with similar diversity of real world infrastructure. Attacks can then be simulated by turning a subset of the virtual machines malicious. These simulations would enable systems to observe attacks as they happen and depending on the specific attack scenario, the testbed can measure the impact on application performance whilst under attack, measure if a subset of compromised nodes can deanoymize users, and other security, privacy, and performance metrics.

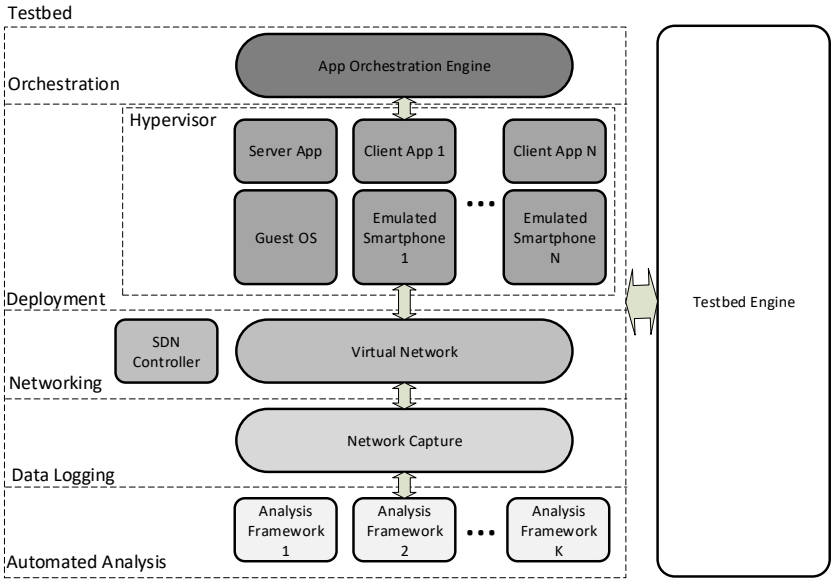
2.3 Privacy Preserving Browsers using Privacy Preserving Networks

The *Tor browser*, *Brave* and other *Onion* browsers use the Tor anonymity network [1] to prevent traceability of communicating parties. They are available for both the *Android* and *iOS* platforms. For *iOS* devices the browsers use the *WebKit* framework, which can override some anonymity features, leaving *iOS* users potentially vulnerable. The *DuckDuckGo* browser promises privacy yet they also have a search engine, which may lead to privacy leaks.

The testbed we propose in this paper can be used to do a comparative study of the browsers on anonymous networks. For example the leakages that might or might not happen due to compulsive use of *WebKit* framework. The economic incentives of *DuckDuckGo* browser against their claims of privacy and how that translates in the network traffic can be tested using our testbed. Our ability to deploy multiple hosts and instances can enable tests to be carried out on the effectiveness of *Tor* against push notifications.

3 Design

The testbed stems from the acknowledgment of the power yet the limit of theoretical models [12]. The requirements of the testbed are captured in Figure 1:



Overview.pdf

Fig. 1: High-level design of the testbed

Deployment. The testbed needs to support the easy deployment of potentially thousands of hosts and services (i.e. the back-end) as well as individual hosts representing users. As well as deploying simple virtual machines, the ability to deploy more modern types of host, such as emulated smartphone environments, is also required. The testbed should provide the functionality to configure machines automatically, including setting machine properties such as hostnames, installing applications to be tested and configuring individual application deployment specific variables such as usernames.

Networking. A realistic virtual network should be deployed for virtual hosts. Complex topologies resembling a real-world deployment can thus be produced. In a real-world setting there may be hundreds of routers and switches involved in the routing of traffic. Each of these, if compromised, becomes a potential point for information leakage to occur and so emulating this environment can provide richer analysis. To support greater flexibility and finer control over the network, software-defined networking is used which allows for the easy deployment of network applications.

Orchestration. Users should be able to automate application functions in order to test at scale without manual intervention. For example, for the contact tracing use cases, users should be able to simulate the broadcast and receive functions required by the application, as well as simulate the interaction between the virtual

hosts. This will be done for diverse platforms as well as for diverse users. Our testbed would include automated navigation within smartphone applications, replaying of network traffic from previous captures or simulated users.

Data Logging. The purpose of the testbed is credible data collection. The diversity of platforms and hosts would mean that the testbed is agnostic and should be able to support data capture from these devices and platforms. The obvious data type to capture is network traffic. As an example, when testing a contact tracing application a tester should be able to send an infection report and the packets containing that report should be captured. As well as network traffic, this can also include live memory captures from virtual hosts and automated screen captures of administrator and user screens.

Automated Analysis. Whilst some users of the testbed will want to perform manual analysis of data captured, for a developer not familiar with privacy analysis frameworks, the testbed should be able to automatically apply such frameworks. For example, by interfacing the data logs with the LINDDUN framework a user will be able to understand the privacy implications resulting out of the trust relationship on the remote entity. The framework includes *hard* and *soft* privacy properties like *unlinkability, undetectability, plausible deniability* and *user content awareness* respectively [9].

4 Prototype Implementation

In order to demonstrate the intended operation of the testbed, we have implemented a prototype. The prototype consists primarily of a command line utility called `kvm-compose`, written in Rust and modeled after the `docker-compose` utility used to manage Docker containers. The `kvm-compose` utility reads a configuration YAML file which specifies which virtual machines should be launched, as well as the network topology to be deployed. Once the configuration file is written, then the testbed can be brought up using a simple `kvm-compose up` command, and shut down using `kvm-compose down`. The utility also allows the user to bring up or tear down specific virtual machines without affecting the rest of the testbed environment.

Listing 1.1: Example machine confguration

```
- name: example1                    # VM Name
  memory_mb: 4096                   # Optional: default 512MiB
  cpus: 4                           # Optional: default 1
  disk:                             # Two variants: cloud_image
    cloud_image:                       or existing_disk
      name: ubuntu_18_04
      expand_gigabytes: 25          # Optional
  interfaces:                       # Connected network interfaces
    - bridge: br0                   #
  run_script: ./script.sh           # Optional: path to a script
  context: ./file.txt               # Optional: path to a file or folder
  environment:                      # Dictionary of arbitary environment variables
    key: value                      # Use /etc/nocloud/env.sh *key* to query
```

Assuming that the disk image supports *cloud-init* at first boot the following will happen:

- The machine `name` (with project prefix) is used as the hostname.
- The SSH public key is injected into the instance.
- File(s) specified in `context` are copied into the `/etc/nocloud/context` directory.
- The `run_script` is executed, with its output log saved into `/etc/nocloud/`.

Virtualization. Virtualization is provided by Kernel-based Virtual Machine (KVM), which is a kernel module for the Linux operating system that allows it to function as a hypervisor. In order to assist in automated machine deployment, `cloud-init` [7] is used, which allows virtual machines to receive a list of data sources (such as URLs or files) with machine deployment information (such as locale, hostname and SSH keys) to be used for that instance.

Networking. Networking is provided using OpenVSwitch (OVS) virtual switches. OVS is used due to its support of software-defined networking (SDN), which allows fine-grained control over the network. The Floodlight SDN controller is used to provide control.

Network Capture. When a test environment has been built (using the `kvm-compose up` command), network traffic can then be collected using the `ovs-tcpdump` utility of OVS [2]. This creates a temporary mirror port on the specified bridge, with traffic from specified ports being mirrored.

5 Reflection and Evaluation with Example Deployments

We used a messaging application *Signal* and a contact tracing application Decentralized Privacy-Preserving Proximity Tracing (DP3T) [6] to test the design considerations of our testbed. The *DP3T* project provides an SDK (Software Development Kit) for both *Android* and *Apple iOS* which is used to communicate with the backend server. It is this library which is used in the official implementations such as *SwissCovid*. *Signal* is a messaging service built using its own custom end-to-end encryption protocol (the *Signal Protocol*), available for a number of platforms on mobile and desktop, designed with a focus on privacy [20].

We have been able to successfully deploy instances of *DP3T* and *Signal* where the virtual machines communicated with external networks to download their dependencies. The SDN controller can be attached to multiple bridges and used to deploy more complex networks. Our testbed successfully captured network traffic from the example projects. A potential improvement could be to integrate the packet capture commands as part of the *kvm-compose* utility to produce a more streamlined experience for the user.

The *DP3T* example demonstrates the use of multiple hosts namely a desktop computer and a mobile phone (*Anbox* and *Android Emulator*). These implementations do not support *Bluetooth* and that can be a limitation for closed binaries (without exposed end-points) where inputs cannot be simulated. However, in our *DP3T* tests, this has not been an impediment as we could simulate inputs. The

UK's NHS COVID-19 contact tracing app implementation is highly coupled to Amazon Web Services (AWS), and as such it is difficult to run within the testbed. *Cloud-native* applications do need standardization for transparent evaluation.

The *kvm-compose* used with *cloud-init* utility makes the testbed easy to deploy and replicate such as the servers for the *DP3T* and *Signal* examples. The level of automation for mobile apps requires further work. While in the *DP3T* example the emulators are installed automatically, it still requires the app to be launched and driven by a user using a window manager. Furthermore, the progress and status (success or failure) during the virtual machine `run_script` phase are not easily accessible, which also impedes extensive automation.

6 Conclusion

The testbed is relevant for developers of systems used by traditional as well as modern hosts in the modern digital economy based on capturing, utilizing and monetizing large-scale information flows. We address at the heart of the information asymmetry that has been characteristic to this eco-system. An entity producing the technologies has more information than the user; the user has no way to verify the claims made by the producer. Our work is a stepping stone towards empowering developers and users.

References

1. `https://www.torproject.org`, accessed June 2021
2. `https://docs.openvswitch.org/en/latest/ref/ovs-tcpdump.8/`, accessed June 2021
3. Acquisti, A., Brandimarte, L., Loewenstein, G.: Secrets and Likes: The Drive for Privacy and the Difficulty of Achieving it in the Digital Age. Journal of Consumer Psychology (2021)
4. Baqer, K., Anderson, R.: Do you believe in Tinker Bell? The social externalities of Trust Transcript of Discussion. In: Revised Selected Papers of the 23rd International Workshop on Security Protocols XXIII - Volume 9379. p. 237–246. Springer-Verlag, Berlin, Heidelberg (2015)
5. Braz, L., Fregnan, E., Çalikli, G., Bacchelli, A.: Why don't developers detect improper input validation? DROP TABLE Papers; –. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). pp. 499–511 (2021). https://doi.org/10.1109/icse43902.2021.00054
6. Busvine, D.: Rift opens over European coronavirus contact tracing APPs (Apr 2020), `https://www.reuters.com/article/uk-health-coronavirus-europe-tech-idUKKBN2221U6?edition-redirect=uk`
7. Canonical: cloud-init - The standard for customising cloud instances, `https://cloud-init.io/`
8. Crispo, B., Lomas, T.M.A.: A certification scheme for electronic commerce. In: Lomas, T.M.A. (ed.) Security Protocols, International Workshop, Cambridge, United Kingdom, April 10-12, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1189, pp. 19–32. Springer (1996). https://doi.org/10.1007/3-540-62494-5_2, `https://doi.org/10.1007/3-540-62494-5\_2`

9. Deng, M., Wuyts, K., Scandariato, R., Preneel, B., Joosen, W.: A privacy threat analysis framework: Supporting the elicitation and fulfillment of privacy requirements. Requir. Eng. **16**(1), 3–32 (Mar 2011). https://doi.org/10.1007/s00766-010-0115-7, https://doi.org/10.1007/s00766-010-0115-7
10. Douceur, J.R.: The sybil attack. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) Peer-to-Peer Systems. pp. 251–260. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
11. Gardiner, J., Craggs, B., Green, B., Rashid, A.: Oops I did it again: Further adventures in the land of ics security testbeds. In: Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy. p. 75–86. CPS-SPC'19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3338499.3357355, https://doi.org/10.1145/3338499.3357355
12. Golomb, S.: Mthematical models - Uses and Limitations. In: Aeronautical Journal (1968)
13. Google, Apple: Exposure notifications: Helping fight covid-19, https://www.google.com/covid19/exposurenotifications/, accessed June 2021
14. Green, B., Lee, A., Antrobus, R., Roedig, U., Hutchison, D., Rashid, A.: Pains, gains and PLCs: Ten lessons from building an industrial control systems testbed for security research. In: 10th USENIX Workshop on Cyber Security Experimentation and Test (CSET 17). USENIX Association, Vancouver, BC (Aug 2017), https://www.usenix.org/conference/cset17/workshop-program/presentation/green
15. Haber, S., Stornetta, W.S.: How to time-stamp a digital document. J. Cryptol. **3**(2), 99–111 (Jan 1991). https://doi.org/10.1007/BF00196791, https://doi.org/10.1007/BF00196791
16. Hoepman, J.H.: Privacy Design Strategies (The Little Blue Book). Radbound University (2019)
17. Lamport, L., Shostak, R., Pease, M.: The Byzantine Generals Problem, p. 203–226. Association for Computing Machinery, New York, NY, USA (2019), https://doi.org/10.1145/3335772.3335936
18. Mathur, A.P., Tippenhauer, N.O.: Swat: A water treatment testbed for research and training on ics security. In: 2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater). pp. 31–36 (2016). https://doi.org/10.1109/CySWater.2016.7469060
19. Nick Triggle: Care.data: How did it go so wrong?, https://www.bbc.co.uk/news/health-26259101, accessed June 2021
20. Signal Foundation: Speak freely, https://signal.org/en/, accessed June 2021
21. Solove, D.J.: The myth of the privacy paradox. George Washington Law Review **89**, 1 (2021)
22. Wang, L., Kangasharju, J.: Real-world sybil attacks in Bittorrent mainline DHT. In: 2012 IEEE Global Communications Conference (GLOBECOM). pp. 826–832 (2012). https://doi.org/10.1109/GLOCOM.2012.6503215
23. Wen, H., Zhao, Q., Lin, Z., Xuan, D., Shroff, N.: A study of the privacy of COVID-19 Contact Tracing Apps. In: Park, N., Sun, K., Foresti, S., Butler, K., Saxena, N. (eds.) Security and Privacy in Communication Networks. pp. 297–317. Springer International Publishing, Cham (2020)
24. Yeh, L.Y., Lu, P.J., Huang, S.H., Huang, J.L.: Sochain: A privacy-preserving DDoS data exchange service over SOC Consortium Blockchain. IEEE Transactions on Engineering Management **67**(4), 1487–1500 (2020). https://doi.org/10.1109/TEM.2020.2976113