Analyzing Communications and Software Systems Security

Siddharth Prakash Rao

A doctoral thesis completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall H304 of the school on 28 August 2023 at 12:00.

Aalto University
School of Science
Department of Computer Science
Abstract

We rely on various communications and software systems where security is critical. Many of these systems have transformed drastically over time with the addition of new features and technologies to accommodate our increasing needs. Unfortunately, such a transformation can introduce new security threats and weaknesses. This dissertation studies security threats and weaknesses in systems that continue to evolve with legacy and modern software components and paradigms.

In this dissertation, we study four different types of information systems: desktop, mobile communications, cloud, and hardware. Our analysis mainly involved building attacks to exploit the vulnerabilities to demonstrate the practicality of our research findings. We uncovered various security issues in each of the systems analyzed. Also, we present various defense and mitigation solutions to address the security issues we found. We discussed our research findings with a wide range of audiences through peer-reviewed publications, responsible disclosure efforts, and by giving talks at various conferences.

The summary of the results is as follows. First, we found insecure use of local communication channels in desktop applications. Second, we discovered several security issues in commercial VPN clients that a network adversary can exploit. Third, we studied mobile communication systems and uncovered security weaknesses of signaling protocols. Also, we present a conceptual framework to model the threats and attacks to mobile networks. Fourth, we demonstrate how adversaries can conduct cross-site scripting attacks by exploiting third-party add-ons of cloud application suites. Finally, we also conduct a human factor analysis to identify usability and security pitfalls faced by software developers when using trusted platform module library APIs. In summary, the contributions of this dissertation include a novel adversary model to study local communication inside a computer, a conceptual framework to study mobile communication systems, the discovery of several new types of security vulnerabilities, and insights into developers' struggles while using security technologies.

Keywords Security analysis, Inter-process communication, cloud-application add-ons, virtual private network, Password managers, cryptocurrency wallets, signaling protocols, threat modeling, trusted platform module

ISSN (printed) 1799-4934 ISSN (pdf) 1799-4942
Location of publisher Helsinki Location of printing Helsinki Year 2023
Pages 213
This dissertation has been made possible due to the support and guidance of many individuals with whom I have had the privilege of interacting during my Ph.D. journey. Some taught me about security and scientific research and helped me stay on track to reach new heights. Some taught me to be emotionally strong when things were not going as intended. All these learnings have definitely helped me become a better researcher and an individual. Thus, I want to give the due credit to those who have been part of this rollercoaster ride.

I am grateful to my Supervisor, Prof. Tuomas Aura, for providing me the opportunity to pursue a Ph.D. with generous funding and research freedom. I would not have opted for a research career without him nudging and inspiring me during my master’s degree. He has patiently guided me since then throughout my doctoral journey, and I know he will continue to do so in the future. He has set a high benchmark for “how to do good research” with his own examples. I shall continue to do good research and pass on the learning to my students hereon.

I thank all my co-authors: Dr. Silke Holtmanns, Dr. Ian Oliver, Dr. Thanh Bui, Dr. Markku Antikainen, Gabriela Sonkeri, Hsin-Yi Chen, and Prof. Janne Lindqvist. My Ph.D. journey would not have been possible without your cooperation. Thanks to Prof. Hanno Langweg and Prof. Jari Porras, pre-examiners of my dissertation, for providing me with valuable feedback that helped to improve the final text. I would also like to thank Prof. Chris Mitchell for agreeing to be the opponent at my public defense.

Special thanks to Thanh for being a fantastic friend and coworker. The Ph.D. journey would have been painful if we weren’t sailing through it together and having fun despite getting frustrated at times. Amidst the numerous silly jokes and so-called “revolutionary” startup ideas, you might have accidentally passed me on some of your wisdom and craftsmanship of getting everything done at lightning speed. Also, thanks to Markku, who initiated the brainstorming sessions with Thanh and me, which eventually helped us collaborate efficiently. The sense of discipline we acquired in these sessions certainly helped us get into top-tier conferences.
I spent most of my time during my doctoral studies at the Aalto University campus in Otaniemi. Thanks to Dr. Sandeep Tamrakar, Dr. Jian Liu, Dr. Aleks Peltonen, and Dr. Mohit Sethi, with whom I had the privilege of sharing the office space and discussing many things about security research. Similarly, the rest of the secure systems group has been a great source of inspiration to learn new research topics in security. Also, thanks to my dear friends Dr. Marius Noreikis, Dr. Gopika Premasankar, Dr. Pranvera Kortoci, and Dr. Maria Montoya Freire for all the fun times we had during and beyond work hours on the campus. I am grateful to Dr. Niina Idänheimo for all her support throughout my research stint at Aalto. Also, thanks for patiently listening to me and guiding me through the challenging phases of my Ph.D. journey. Thanks to all the support staff at the Department of computer science, Aalto University.

I want to thank all my colleagues at Nokia Bell Labs for supporting me during various stages of my Ph.D. journey. In particular, I am grateful to Silke, who introduced me to the world of telco security. Thanks to Gabriela, one of the best research companions and colleagues I could ever have. Gaby, thanks for patiently teaching me about trusted platforms and listening to my rants. Special thanks to Dr. Yoan Miche, my line manager at Nokia Bell Labs. He offered me an opportunity to work in his team right when I was finishing my work at the University. He provided a comfortable work environment to conduct research without boundaries. He has been an excellent mentor and friend who has gone beyond his duty as a manager by providing emotional support and boosting my self-confidence in every possible way. Thanks to Hsin-Yi, my first full-time intern, whose master’s thesis came in handy to bring the Bhadra framework to practice.

During 2016-17, when I was still considering whether to pursue a Ph.D., I had the opportunity to try something different that substantially changed how I define technology research. As a Ford-Mozilla Open Web Fellow at European Digital Rights (EDRI), I worked with lawyers, policymakers, artists, activists, civil society organizations, corporations, and many others outside academia. All my interactions and learning during the fellowship have strongly influenced the choice of my research topics thereon. Thanks to colleagues at EDRI, Mozilla Foundation, and Ford Foundation for the opportunity to work on projects with a social impact. Special thanks to Joe McNamee, Kirsten Fidler, Vanessa Rhinesmith Dr. Mike Brennan for the mentorship and for making the fellowship a very memorable experience. Thanks to all the open web fellows from my cohort — Matt Mitchelle, Etienne Maynier, Berhan Taye, Stefania Paola, Eireann Leverett, Jen Helsby, and Suchana Seth — for the lifelong friendship and inspiration. Special thanks to Romina Lupseneanu, Zarja Protner, and Aimilia Givropoulou for being my best friends and excellent coworkers who gracefully helped me handle my chaotic life in Brussels.

My friends in India have been a phenomenal support system remotely
and during my visits to India. Thanks to Bhargavi Ramakrishna (my mother-in-law), all my cousins and the rest of the family members, Swathi Ramakrishna (my sister-in-law), and little Ameya for our shared happy moments. Thanks to Spandan, Prathik, Dr. Vinay, Shivapramod, Naresh, Shrestha, Anusha, Abhilash, Seema, Dr. Harini, and Dr. Praveen for cheering me up always and making me laugh my heart and lungs out. Thanks to Venkatesh, Navneeth, Geeth, and Somani for sensing my lows and highs without being explicitly told and reciprocating accordingly. Thanks to Mufthi, Prajin, Sunil, Sanjay, Nitin, Kitty, Anul, Roohi, and Chaitanya for keeping me grounded.

Living away from home had started taking a toll on me at some point. The Finland Kannada Association (FINKA) rescued me by becoming my second family. Thanks to each one of you for saving me from falling severely homesick. Special thanks to Nanda Kumar, who found ways to cheer me up with good music and profound discussions. Thanks to Dr. Kiran Hasygar, Dr. Ashwini Nagaraj, Dr. Yashavanthi Mysore, and Dr. Raghavendra Mysore for the peer support. Also, thanks to Narahari and Rakshith for the evening walks and laughter sessions, which helped me sail through the most challenging phase of my Ph.D. journey.

My family has been my biggest support throughout this journey. The idea of pursuing a doctoral degree was instigated by My dad Prof. Prakash Rao Barkur. He taught me to dream big, aim high, and put my heart and soul into my work. A zillion thanks goes to my dearest mother, Vibha Prakash Rao. Her endless prayers and wishes have often done wonders in all aspects of my life, and the Ph.D. journey is no exception. Pappa and Mamma, I would not step outside my tiny cocoon in Thirthahalli if you did not encourage me to get comfortable outside my comfort zone.

Last but not least, my deepest gratitude to my soulmate Savi Ramakrishna. You have showered me with unconditional love and affection despite being devoid of my attention and time due to my never-ending research musings. Thanks for offering me a shoulder to lean on whenever I am low. Thanks for being part of every little celebration. Thanks for being supportive of everything I do on professional and personal fronts. And a gazillion thanks for creating Soham, the best gift I could ever ask for!

Espoo, June 26, 2023,

Siddharth Prakash Rao
Contents

Preface 3

Contents 7

List of Publications 9

Author’s Contribution 11

List of Figures 17

List of Tables 19

1. Introduction 21
   1.1 Scope and research goals . . . . . . . . . . . . . . . . . . . . . . 23
   1.2 Research approach and methodology . . . . . . . . . . . . . . 25
   1.3 Research contributions . . . . . . . . . . . . . . . . . . . . . . . 28
   1.4 Structure . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31

2. Inter-process Communication 33
   2.1 Background . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
   2.2 Vulnerabilities in desktop applications . . . . . . . . . . . . 35
   2.3 Vulnerabilities in cryptocurrency wallets . . . . . . . . . . . 38
   2.4 Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39

3. Virtual Private Networks 43
   3.1 Background . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
   3.2 Vulnerabilities in commercial VPN clients . . . . . . . . . . 44
   3.3 Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47

4. Cellular Communication 49
   4.1 Background . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49
   4.2 Vulnerabilities in mobile network signaling . . . . . . . . . 51
   4.3 Bhadra — a domain-specific threat modeling framework . 53
   4.4 Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54
5. API-based Communication
   5.1 Background .............................................. 57
   5.2 Vulnerabilities in cloud applications add-ons .............. 58
   5.3 Usability of trusted platform module library APIs .......... 61
   5.4 Discussion ............................................... 63

6. Lessons Learned ............................................. 65
   6.1 Need for a standard channel for vulnerability reporting ... 65
   6.2 Tackling software dependencies ............................. 68
   6.3 Empowering developer communities .......................... 68
   6.4 Systems security as a multidisciplinary problem space ... 69

7. Concluding Remarks .......................................... 71

References ...................................................... 73

Publications .................................................... 81
This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.


V Silke Holtmanns, Siddharth Prakash Rao, and Ian Oliver. User Location Tracking Attacks for LTE Networks Using the Interworking Functionality. In *2016 IFIP Networking Conference (IFIP Networking)* and
List of Publications


Author’s Contribution

Publication I: “Man-in-the-machine: Exploiting Ill-secured Communication Inside the Computer”

I was one of the two principal authors of this publication. Viswanathan Bojan, who did his master's thesis in our research group, conducted a security analysis of password managers that comprise a desktop application and browser extension. He considered mainly threats from malware that has the same access rights as the victim and can intercept the communication between the software components. The possibility of a similar interception from a simultaneous user session emerged during the discussion between Thanh Bui and me. Together with the other authors, I extended Bojan’s work by generalizing the problem and finding more examples of password managers and other applications. I also contributed to the literature survey and coordinated the responsible disclosure of the discovered vulnerabilities to the affected vendors. All the authors contributed to writing the paper.

Publication II: “Pitfalls of Open Architecture: How Friends Can Exploit Your Cryptocurrency Wallet”

I was one of the two principal authors of this publication. This publication extends the results from Publication I. Thanh Bui and I analyzed several cryptocurrency desktop wallet applications under the Man-in-the-Machine threat model and discovered similar vulnerabilities. I also coordinated the responsible disclosure. Markku Antikainen helped with formulating the problem statement. Tuomas Aura gave feedback, supervised the research and helped with editing the final text. All authors contributed to writing the paper.
Publication III: “Client-Side Vulnerabilities in Commercial VPNs”

I was one of the two principal authors of this publication. During the research on communications between mobile operators, I observed the lack of clear guidelines on secure configurations for VPN protocols. On the other hand, Thanh Bui observed that many VPN providers use insecure configurations for L2TP/IPSec protocol despite knowing the security issues for years. At the same time, Thanh Bui and I were supervising student projects for a university course, which involved setting up a VPN connection, and most students could not configure it securely at the first attempt. Therefore, we decided to conduct an in-depth analysis of VPN client configurations by combining our individual observations. Markku Antikainen helped with formulating the problem statement. Thanh Bui and I conducted the analysis. I also coordinated the responsible disclosure. Tuomas Aura gave feedback, supervised the research and helped with editing the final text. All the authors contributed to writing the paper.

Publication IV: “Unblocking Stolen Mobile Devices Using SS7-MAP Vulnerabilities: Exploiting the Relationship between IMEI and IMSI for EIR Access”

I was the principal author of the publication. This work builds on my master's thesis, which surveyed attacks that rely on the Signaling System 7 (SS7) protocol used in the mobile communication backend. I wrote the publication text with editorial help from Tuomas Aura. Silke Holtmanns supervised me and discussed the discovered issues with the standardization bodies. Ian Oliver provided expert advice.

Publication V: “User Location Tracking Attacks for LTE Networks Using the Interworking Functionality”

I was one of the two principal authors. The initial idea is based on the discussions and observations between the two main authors. I improved the initial idea, conducted in-depth research, and wrote the paper. Silke Holtmanns contributed to disseminating the results to the standardization bodies and other relevant parties. Ian Oliver provided expert advice.

I was the principal author of the publication and conceived the idea of a threat modeling framework for mobile communication systems. I conducted the literature survey and categorization of threats, designed the framework, and wrote the draft. Hsin-Yi Chen developed the threat modeling tool that is used in the analysis. Tuomas Aura supervised the research and helped with editing the final text.

Publication VII: “XSS Vulnerabilities in Cloud-Application Add-Ons”

I was one of the two principal authors of this publication. Together with Thanh Bui and Markku Antikainen, I conceptualized the threat model. Thanh Bui and I analyzed the add-on and the back-end architecture to understand the impact of XSS attacks on cloud applications. I also examined the permission-based access control of the add-ons. I also coordinated the responsible disclosure. Tuomas Aura gave feedback, supervised the research and helped with editing the final text. All authors contributed to writing the paper.

Publication VIII: “Usability and Security of Trusted Platform Module (TPM) Library APIs”

I was one of the principal authors of this publication. I conceived the idea of conducting a usability and human factors analysis of the TPM library APIs, and Gabriela Limonta provided the TPM expertise. I contributed to the design of the questionnaires, interviews, and usability experiments. Also, I wrote the recommendations for improving the TPM library API documentation and software. Gabriela and I conducted the interviews, analyzed the data, and wrote the paper. Janne Lindqvist gave feedback and supervised the research.
During my doctoral studies, I also worked on the following publications, which are not included in this dissertation.


List of Figures

2.1 Man in the Middle (MitM) vs Man in the Machine (MitMa) 34
2.2 MitMa attacks on password managers . . . . . . . . . . . . . 36
4.1 Interworking between 2G/3G and 4G core networks . . . . 52
4.2 Overview of various kinds of adversaries in mobile networks 54
5.1 Variants of the XSS attack with vulnerable add-ons . . . . 59
5.2 Architectures of cloud applications add-ons . . . . . . . . . 60
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Publication-wise mapping of research scope and goals</td>
<td>24</td>
</tr>
<tr>
<td>1.2</td>
<td>Mapping of communication channel, publication and target type and systems.</td>
<td>26</td>
</tr>
<tr>
<td>2.1</td>
<td>Summary of vulnerabilities in desktop applications (in 2018)</td>
<td>37</td>
</tr>
<tr>
<td>2.2</td>
<td>Summary of vulnerabilities in cryptocurrency wallets (in 2019)</td>
<td>39</td>
</tr>
<tr>
<td>3.1</td>
<td>Summary of VPN client vulnerabilities (in 2019)</td>
<td>46</td>
</tr>
<tr>
<td>5.1</td>
<td>Summary of usability and security pitfalls of tpm2-tools library APIs</td>
<td>62</td>
</tr>
</tbody>
</table>
1. Introduction

Our reliance on technology has increased drastically over the past decades. Technology has become an integral part of our day-to-day lives, and consequently, most of our daily activities involve digital products and communications. Many of us have witnessed and appreciated how different technologies evolve. However, the increased complexity of communications and software systems that accumulates along the technology evolution remains hidden from the public view. On the one hand, the complexity of the software code base increases because of the continuous introduction of new features or technologies and software development methods. On the other hand, the complexity of the entire system increases due to the inter-dependency between legacy and modern components and the diversity of co-existing technologies. The obvious problems caused by such accumulated complexity are reduced maintainability and reliability. Another often overlooked yet equally important problem is that the complexity creates barriers to achieving security. Thus, analyzing these security issues and discussing the potential solutions becomes even more crucial. This dissertation contributes to the knowledge of the various security threats and weaknesses in communications and software systems, with the hope that the increased understanding leads to new and robust solutions.

This dissertation consists of eight original publications and this compendium part. The individual publications included in the dissertation are about several different information systems, where the potential adversaries and the research methodology are also different. However, an aspect that is common across all the publications and binds them together is that all the publications are about security issues in communications between the components in distributed systems. We wanted to understand security issues in the broad spectrum of communication channels that exist in modern computing systems. The publications cover inter-process communication, virtual private network communication, 3GPP-based mobile network communication, and API-based communication. In this compendium, we dedicate one chapter for each communication type and present a summary of the publications that analyze security issues in
that type of communication.

The other common and binding factor is that all the publications, except Publication VIII, follow \textit{adversary-centric threat models} to understand the adversary and uncover various security issues. In adversary-centric threat models, a system’s security is assessed from the viewpoint of an adversary by making pragmatic assumptions about the adversary’s environment and resource access, goals, and capabilities \cite{36}. We implement attacks to exploit the vulnerabilities in the systems to demonstrate the practicality of our research findings. The findings include software vulnerabilities as well as architectural or design weaknesses that lead to security vulnerabilities. On the one hand, this approach allowed us to discover new types of adversaries; on the other hand, it helped us investigate new applications and environments under the known adversaries. We also provide best-practice recommendations and discuss various defense and mitigation solutions to fix the security issues we discovered. We believe that an \textit{asset-centric threat model}, where the analysis is done from the viewpoint of a defender to secure the assets, would have yielded similar results. However, the research focus could have shifted towards theoretical threats and less realistic attacks.

The research for this dissertation involved analyzing a large number of software products and services. We also analyzed widely deployed protocols in the mobile communication networks. A prominent type of security issue found in our analyses is related to missing or weak authentication of endpoints that allows an adversary to conduct client or server impersonation and man-in-the-middle attacks. One of the common reasons is the lack of motivation and action by software vendors and service providers who continue to support insecure technologies. We also observed a lack of security awareness about adversarial capabilities among software architects and developers. Our work strives to create security awareness by documenting common security mistakes and highlighting the missing or forgotten best practices. We conducted responsible disclosure of the discovered vulnerabilities and engaged with software developers to discuss how they would address the security issues. Such discussions also helped us curate the defense strategies we presented in the publications. While our peer-reviewed publications and vulnerability disclosures are sufficient to spur discussions among academic research and developer communities of the specific technologies, their reach may be limited only to those communities. Thus, to create public awareness of the security issues, we coupled our academic-style publications with engagement in engineering and public venues with a broader audience. These efforts included giving talks at hacker venues (namely, DEF CON \cite{28}, Black Hat \cite{74}, Troopers \cite{95}, and Disobey \cite{29}), discussing the implications of some of our findings with reputed global news media like \textit{The Guardian} \cite{72, 71, 24}, and research collaborations with civil society organizations \cite{76}.
1.1 Scope and research goals

The overall objective of this thesis is to provide improved understanding of the factors and processes that influence communications and software systems security. The publications included in this thesis analyze various real-world systems from the points of view of system design, surrounding threat landscape, implementation, and human factors. Table 1.1 summarizes the relation of the included publications to these system aspects.

**System design considerations:** *System design* the process of gathering requirements and, based on them, defining the system architecture, components and processes. An integral part of the system design is defining the communication channels and how the communication takes place between the components and with other systems. System design typically involves careful planning in accordance with design methodologies or frameworks, technical specifications and standards, and legal regulations. Also, the design must satisfy security requirements and protect the system against well-known threats. Failing to do so would lead to design flaws that can severely affect the system’s reliability. These flaws typically include mistakes, misconceptions, and inefficient planning at an early stage of the technology lifecycle. Unfortunately, the system design weaknesses have not received as much attention as the software implementation or code-level bugs [15, 97]. While fixing implementation security bugs can be done with a patch or software update, design flaws often require substantial changes to many system elements that may be infeasible to implement.

We reviewed system design considerations by analyzing the technical specifications standards and software documentation. In particular, we reviewed cellular standards in Publications IV and V, various software documentation in Publication VII, and trusted hardware standards in Publication VIII. During the analysis, we looked for flaws that may have stemmed from the assumptions and choices made by the designers. This analysis approach helped us investigate the root causes of security issues at a much deeper level than if we only analyzed the current implementation. Furthermore, it provides us a bird’s-eye view of adversarial capabilities and threats to the system as well as insight into alternative architectural designs as defense solutions.

**Threat landscape:** *Threats* are potentially bad events caused by adversaries with malicious intent. Adversaries are the entities who would like to cause harm to a system. On the other hand, *threat landscape* refers to the entirety of both observed and yet unrealized threats. Mapping all the threats, e.g., through threat modeling [117], to formulate the threat landscape helps in understanding security weaknesses and serves as a prerequisite to designing defensive strategies [84]. Threat landscapes change as technology and the systems based on it evolve. For example,
introducing a new feature to an existing system, technology upgrades and transformations, new applications for the technology, availability of restricted hardware or software in the public domain, or changes in legal regulations may introduce new threats. Therefore, keeping track of the threats and changes to the threat landscape must be a regular activity from a systems security perspective. We put emphasis on studying a system’s changing threat landscape because it provides insight into new attack surfaces and changes in adversarial capabilities. This emphasis is realized in our research in two forms. Firstly, in Publication VI, we capture the current threat landscape of the mobile communication system by analyzing threats to its components. This complex system contains both legacy and constantly evolving components. Our research reviews its current threat landscape from the perspective of relevant adversaries at the time of writing the publication and provides a conceptual framework for threat modeling. Secondly, in Publication I and VII, we introduce novel adversary models to capture the changed threat landscape in specific systems.

**Implementation:** Software implementation can introduce its own flaws or reflect poor design decisions, and our research explores both. We attempted to discover new types of vulnerabilities and understand their root causes. Also, we looked for known vulnerabilities in the software implementations. We contribute to the ongoing efforts to create security awareness among developers by highlighting common security mistakes and best practices that have not previously received sufficient attention. We analyzed the software implementations of desktop applications such as password managers, cryptocurrency wallets, and VPN clients in Publications I, II, and III. In Publication VII, we analyzed the implementations of cloud applications and their add-ons.

**Human factors:** Traditionally, security research attempts to analyze the technical properties of systems to solve security issues. However, recent research works have also attempted to analyze security issues arising from the interaction between users and systems [116, 56, 70]. These research works explore human factors influencing security-related decisions, such as usability, mental models, and user experience and perceptions [16, 12, 11, 92, 55]. In Publication VIII, we look into some of these factors by analyzing

---

**Table 1.1.** Publication-wise mapping of research scope and goals

<table>
<thead>
<tr>
<th></th>
<th>PI</th>
<th>PII</th>
<th>PIII</th>
<th>PIV</th>
<th>PV</th>
<th>PVI</th>
<th>PVII</th>
<th>PVIII</th>
</tr>
</thead>
<tbody>
<tr>
<td>System design</td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Threat landscape</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>Human factors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
</tr>
</tbody>
</table>
the problems around how expert users, particularly software developers, interact with and build secure systems. We focused on software developers because their human errors can impact many systems at once.

Threat modeling is another area where human factors, such as the understanding of the security analyst, play a crucial role [117, 44]. In Publication VI, threat modeling is used for describing and characterizing known threats and attacks to facilitate human communication. The publication provides a conceptual framework for security analysts to explain threats on a conceptual level that abstracts away a lot of the technical detail.

1.2 Research approach and methodology

We followed methodologies that are well-established in information systems security research. On the high level, we analyzed artifacts such as protocols, testbed networks, and published software to identify security issues and potential solutions. The findings from the analysis were used for design generalization, i.e., to infer design principles for an entire class of information systems. The following is a detailed summary of the research approach and methodology.

Generic outline: In the course of the research, we analyzed four types of information systems that are part of our day-to-day lives: desktop applications, mobile communication, cloud-based productivity applications, and hardware technology that is integrated into communication devices. In each type, we chose examples where the security of communication channels is critical. We selected both widely-used information systems and a variety of technologies to find vulnerabilities with practical impact and to gain a broad understanding of the technical issues. The target systems include widely used consumer software applications. We also investigated the protocols and architecture of global mobile communications networks. Table 1.2 summarizes how the included publications relate to these systems.

Understanding the role of the adversary is necessary for discovering security issues and providing solutions to them [36]. So, we first defined a threat model for each type of system. Then, we considered known security issues and current attack trends to form a picture of an adversary’s environment, resource access, goals, and capabilities. We then analyzed the systems from the adversarial viewpoint and experimentally evaluated any potential security weaknesses by building proof-of-concept attacks. We discovered design and implementation weaknesses and highlighted overlooked adversary models. In Publication VIII, we did not explicitly define a threat model. Instead, we designed tasks in which the study participants were expected to come up with a threat model or make assumptions about
<table>
<thead>
<tr>
<th>Communication channel</th>
<th>Publication</th>
<th>Target Type</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication inside a computer</td>
<td>PI</td>
<td>Desktop applications</td>
<td>Password managers</td>
</tr>
<tr>
<td></td>
<td>PII</td>
<td></td>
<td>Cryptocurrency wallets</td>
</tr>
<tr>
<td>Virtual Private Networks</td>
<td>PIII</td>
<td></td>
<td>VPN clients</td>
</tr>
<tr>
<td>3GPP-based cellular networks</td>
<td>PIV</td>
<td>Mobile communication systems</td>
<td>2G and 3G</td>
</tr>
<tr>
<td></td>
<td>PV</td>
<td></td>
<td>3G → 4G</td>
</tr>
<tr>
<td></td>
<td>PVI</td>
<td></td>
<td>2G, 3G, 4G</td>
</tr>
<tr>
<td>API-based communication</td>
<td>PVII</td>
<td>Cloud applications</td>
<td>Add-ons</td>
</tr>
<tr>
<td></td>
<td>PVIII</td>
<td>Hardware interfaces</td>
<td>TPM libraries</td>
</tr>
</tbody>
</table>

adversarial capabilities. Then, we analyzed the solutions to the tasks to evaluate whether the participants had correctly identified and resolved any potential threats.

Analyzing with an adversarial viewpoint carries scientific value and real-world impact if the lessons learned are shared with the research community [21]. Therefore, in addition to reporting individual vulnerabilities, we put most effort on providing meaningful insights on security issues that affect an entire industry or area of technology. Also, we checked whether the vulnerabilities found in one system were reproducible in other systems of the same type. Industrial security certification standards describe similar analysis processes and steps, but their goals differ from our research. For example, Common Criteria for Information Technology Security Evaluation (CC) [6] and ISO/IEC 27000-series [65] aim to find and prevent commonly known vulnerabilities and certify that the analyzed system is free from such vulnerabilities. On the other hand, our research aims to extend the analysis also to find previously unknown types of vulnerabilities and solution to them. Our research has produced knowledge of new classes of vulnerabilities which should eventually become part of the security evaluation of the certification standards.

**Analysis and responsible disclosure:** In the course of the research work behind Publications I, II, III, and VII, we analyzed 30 desktop ap-
applications, 30 VPN clients, and 300 cloud application suite add-ons. For each security issue found, we conducted responsible disclosure to report our findings to the relevant parties. We reached out to various product and OS vendors, service providers, and open-source code repository maintainers. We confirmed with them that the findings are reproducible. Also, we utilized the responsible disclosure process as an opportunity to discuss potential solutions.

In Publications IV and V, we took a slightly different approach to analyze the standard telephony protocols for signaling between mobile operators. First, we identified security issues by a rigorous analysis of 3GPP standard specifications. Then, we evaluated them in an experimental setup to ensure that the findings translate to practical attacks. Experimenting on real mobile networks was not possible due to safety and data security concerns. Instead, we confirmed our findings with experts from the mobile communication industry and standardization bodies who helped to gauge their impact on real networks. Unfortunately, there is little public information about the exact network configurations, firewalls or other defense solutions deployed by mobile operators in real networks. Therefore, our work contributes to the 3GPP standards and alerts mobile operators about potential vulnerabilities which they need to check themselves. The responsible disclosure included reporting the findings to the GSMA Fraud and Security Group (FSAG\textsuperscript{1}) and to the 3GPP Service and System Aspects Working Group 3 (SA3 - Security\textsuperscript{2}). One of the co-authors of Publication IV and V coordinated this work.

In Publication VI, we surveyed the threat landscape in mobile communications networks and presented a conceptual framework for threat modeling. We analyzed known attacks against mobile communications from research and professional literature and organized them into attack phases, tactical objectives, and techniques. We presented our work to GSMA and MITRE and advocated for forming a dedicated working group to extend the framework.

In Publication VIII, we conducted a qualitative study with Trusted Platform Module (TPM) developers using mixed methods, i.e., task analysis and cognitive interviews. We identified common use cases of TPM, reviewed the available APIs, conducted a literature survey of the prior art, and combined them to design tasks and questionnaires for our participants. We also involved the participants in a follow-up interview to understand their experiences, perceptions, and opinions about the APIs. We conducted thematic analysis and code analysis to identify themes and common coding patterns that give an overview of the usability and security pitfalls of the tpm2-tools\textsuperscript{3} library. Based on these results, we provided concrete recom-
recommendations for the library documentation and software. We reported the findings and recommendations to Tpm.dev, a community forum for TPM developers and library and specification contributors.

Repeatability of the research: We diligently documented the analysis procedures and reported them in individual publications such that future work could repeat and extend the research methodology. However, we analyze living software and evolving threats, and reproducing the same vulnerabilities in the future is usually not interesting. Instead, the analysis process and methods should be repeatable, so that they can be applied to new and improved target systems to ensure that the same vulnerabilities do not reoccur.

1.3 Research contributions

This section summarizes the research contributions in each of the peer-reviewed publications included in this dissertation.

Publication I introduces a novel adversary model called the Man-in-the-Machine (MitMa), where a non-privileged user of a multi-user computer system intercepts the communication occurring inside the computer. More specifically, an adversary from its user session exploits the Inter-Process Communication (IPC) channels between the processes of other users who are on the same machine but have a different user session. Although interception or tampering of communication is a standard adversary model in networked communication, local communication is assumed to be immune to such attacks. We show that certain types of IPC are vulnerable to client and server impersonation, and that such flaws exist in widely used security-critical software applications such as password managers and hardware tokens. The lesson from our work is that developers should treat local communication security with the same prudence as network communication. With a new adversary model and discovery of various vulnerabilities, Publication I contributes to the threat landscape and implementation objectives of the thesis.

Publication II extends the security analysis of desktop applications under the MitMa adversary model. It analyzes the security of the Remote Procedure Call (RPC) interface of the cryptocurrency wallet desktop applications. These interfaces are, in fact, used for inter-process communication on the localhost. The provision of an RPC interface promotes the development of other blockchain-based applications by allowing them to access the wallet’s functionality. However, adversaries like MitMa can exploit such interfaces. As demonstrated in this work, malicious processes created by other authenticated but unprivileged users of a multi-user system

4https://developers.tpm.dev/
can impersonate the communication endpoints of the RPC channel and steal funds from the wallet. We analyzed several kinds of desktop wallet applications that offered varying protection levels to the RPC interface and showed that none of those protection measures were effective against MitMa. This publication shows that attacks from local adversaries have not received enough attention from the security research community. Publication II thus contributes to the threat landscape and implementation objectives of the thesis.

Publication III analyzes desktop applications from the viewpoint of a traditional network adversary. In particular, this work analyzes the implementation and configuration of commercial Virtual Private Network (VPN) desktop clients. We studied how these VPN clients set up VPN tunnels and how VPN service providers instruct the end-users to configure their client software. By analyzing 30 widely used VPN services, we discovered various security flaws in VPN clients that allow adversaries to strip off traffic encryption or bypass server authentication, thereby intercepting the victim’s network traffic. We found that these flaws were due to simple configuration mistakes, poor instructions for the users, incorrect developer guidelines, insecure default values, and failure to disable broken legacy features. Many of these flaws are unnecessary, trivial, and repeated, which indicates a serious lack of security awareness across the commercial VPN industry. In this study, most of the results are about the incidence of known vulnerabilities, contributing to the implementation objective of the thesis.

Publication IV looks at a completely different system — the mobile communication networks based on the 3GPP standard specifications. We present a Signaling System Number 7 (SS7) based attack to manipulate the Equipment Identity Register (EIR), the entity responsible for tracking and blocking stolen mobile devices. The attack exploits the relationship between the International Mobile Equipment Identity (IMEI) and International Mobile Subscriber Identity (IMSI) to override denylisted or stolen mobile devices to an allowlist. With the analysis of mobile communication backend networks and potential flaws in the network architecture, Publication IV contributes to the system design objective of the thesis.

Publication V analyzes inter-protocol and inter-generation signaling in mobile communication networks. These networks achieve uninterrupted services to mobile users by maintaining interoperability between mobile operators and backward compatibility with older technology generations using methods standardized by 3GPP. However, as our work shows, the interoperability and backward compatibility achieved by such methods may introduce new security threats. In particular, we demonstrate how an adversary with access to an operator's backend network can misuse Inter Working Functionality (IWF), a method to translate signaling messages between the 4G Long Term Evolution (LTE) and 2G and 3G networks. This work highlights the pattern that vulnerabilities in old technology
may creep to newer systems through backward compatibility features. Publication V therefore contributes to the \textit{system design} objective of the thesis.

**Publication VI** presents \textit{Bhadra}, a domain-specific conceptual framework for modeling threats and attacks against mobile communication systems. Inspired by the MITRE ATT&CK framework,\footnote{https://attack.mitre.org/} we built our framework by surveying publicly known attacks against mobile communications and organizing them into attack phases, tactical objectives, and techniques. The Bhadra framework provides a structured way to analyze and communicate security events on a level that abstracts away the technical details while still providing meaningful insights into the adversarial behavior. The paper includes concrete case studies of applying the framework. Publication VI contributes to the \textit{threat landscape} and \textit{human factors} objectives of the thesis.

**Publication VII** analyzes the security of another type of system -- \textit{cloud applications}. Modern cloud applications implement much of their functionality as independent services that are loosely coupled to the core service through APIs. Many cloud-application vendors open their APIs to third-party developers to extend the functionality of their applications. The features implemented with these APIs are called \textit{add-ons}. We studied how add-on services process untrusted user input by analyzing architecture designs and security mechanisms of cloud applications. We discovered two ways to inject attack payloads and conduct XSS attacks against cloud-application users through vulnerable add-ons. Through an empirical analysis of add-ons in the wild, we discovered that a significant percentage of add-ons are vulnerable to XSS attacks. We also discussed secure design choices for cloud application architecture and best practices for add-on developers. Publication VII contributes to the \textit{system design} and \textit{threat landscape} objectives of the thesis.

**Publication VIII** continues the analysis of APIs however from human-computer interaction (HCI) point of view. Trusted Platform Modules (TPMs) provide a hardware-based root of trust and secure storage. Software developers can interact with a TPM and utilize its functionality with standard APIs that various libraries have implemented. We explored the usability and security pitfalls when software developers use \texttt{tpm2-tools}, a widely used TPM library. Publication VIII demonstrates that the \texttt{tpm2-tools} APIs are not user-friendly based on the analysis of developer input collected through tasks and interviews. For this, we built a platform for studying TPM-related tasks. The platform supports all major TPM libraries and works right out of a browser \cite{4}. Our findings support those of past studies \cite{56} and provide new insights for the trusted computing community for usable secure API development. Publication VIII contributes to the \textit{system design} and \textit{threat landscape} objectives of the thesis.
design, implementation and human factors objectives of the thesis.

1.4 Structure

The rest of the compendium extends the discussion in the introduction. All seven original publications are included at the end. The compendium is organized as follows:

We present a summary of the publications that are grouped into four chapters, where each chapter is dedicated to one type of communication channel. Chapter 2 discusses the security issues in local communication channels and summarizes the findings from Publications I and II.

Chapter 3 presents the security issues in Virtual Private Networks and summary of the findings from Publication III.

Chapter 4 discusses the security issues in mobile communication and summarizes the findings from Publications IV and VI.

Chapter 5 presents the security in API-based communication channels of cloud applications and TPM. This chapter summarizes the findings from Publications VII and VIII.

Chapter 6 presents common observations and lessons learned. Finally, we present the concluding remarks in Chapter 7.
2. Inter-process Communication

In this chapter, we summarize the findings from Publications I and II, which analyzed the security of local communication between processes that run inside a computer. We presented a novel adversary model called Man in the Machine (MitMa) and found several vulnerabilities in security-critical desktop applications.

2.1 Background

In traditional network security threat models, the communication endpoints — e.g., user devices and remote servers — are trusted, but the network where they communicate is untrusted. The presence of adversaries in an untrusted network makes the communication endpoints susceptible to Man in the Middle (MitM) and impersonation attacks. However, not all communication goes over computer networks in the traditional sense. For example, communication between different processes of desktop software applications that run inside a computer falls into this category. Our work studies this previously overlooked local communication by applying traditional network security analysis methods to it.

Client-server web applications are examples of software that is divided into multiple components, such as User Interface (UI) on the frontend and server and database on the backend. Sometimes, the backend is not remote across the Internet but local on the same computer. Many native desktop applications follow a similar software design pattern where the native UI component connects to a server or database that may be on the same computer rather than remote across a network. A special type of software component is the browser extension, which is a software module integrated into the web browser. Some browser extensions act as a frontend or agent for desktop software, such as password manager. In that case, they need to communicate with the desktop software on the local machine. At the OS level, these components run as individual processes that communicate with each other over Inter-Process Communications (IPC), or in some
cases, over Remote Procedure Calls (RPC). Since both endpoints of these communication channels lie within the personal computer, they appear to be operating in a safe environment free of external security threats. Nevertheless, we argue that local communication could have security vulnerabilities similar to those in communication that crosses a network.

Both personal computers (PC) and the desktop applications running in them are assumed to be personal or single-user systems. As a result, security engineering efforts have focused on threats from the outside, e.g., from the Internet. In reality, most PCs are multi-user systems that are shared or could be accessed by more than one user, such as family members, coworkers, or guest users. All these secondary users who can log into the PC and execute software in it are potential adversaries. The security research and engineering communities have a somewhat ambiguous attitude towards threats from such local adversaries. On the one hand, modern desktop operating systems have taken measures to protect users from local adversaries when multiple users share a PC. For example, the Shared PC mode in Windows 10 enforces temporal separation of users and prohibits multiple sessions running in parallel [2]. On the other hand, many security experts consider the threats from local adversaries to be unrealistic scenarios or unsolvable problems. They tend to believe that local adversaries are tech-savvy and powerful users, who can gain administrative access with privilege escalation techniques [5, 67]. This is especially the case when the adversaries have physical access to the device. Nevertheless, we demonstrate in Publications I and II that local adversaries, such as coworkers and family members, with minimal technical expertise can be a real threat that requires attention from software developers. In particular, we show how such overlooked adversaries can intercept local communication inside a computer without any privileged access.

We refer to our adversary model as Man in the Machine (MitMa) because it is similar to MitM attacks but takes place inside a computer, as shown in Figure 2.1. Here, a local adversary with non-administrator access, such as a normal user or guest, has the ability to keep non-privileged processes
running in the background. These processes can then exploit the local communication of other users on the same machine. Unlike malware that runs with the same privileges as the victim and tries to escalate to administrator privileges, the MitMa adversary impersonates the local communication endpoints from a different login session than the victim’s session. While Windows PCs have not been designed to support multiple simultaneous users, features such as fast user switching\(^1\) make it possible to leave a user or guest session in the background and resume it later. The background sessions continue to have running processes that can act as the MitMa attacker. The same results can also be achieved on Linux and macOS, where processes can be kept alive even after the user logs out, e.g., with the `nohup` command. If SSH or remote desktop sessions are enabled, the background login session or processes can be created remotely.

Our work introduces the MitMa adversary model and demonstrates its relevance by finding vulnerabilities in security-critical desktop applications. In Publication I, we analyzed widely-used password managers, two-factor authentication, and other applications that had not given sufficient attention to securing the IPC channels. In Publication II, similar issues were found in popular cryptocurrency desktop wallet applications that use RPC channels for local communication. In the rest of this chapter, we provide a brief overview of our findings.

### 2.2 Vulnerabilities in desktop applications

Publication I presents a security analysis of desktop applications. In particular, we analyze password managers with two software components that run on the same computer. The first component is a native desktop app that manages the password vault. The vault stores the user’s login credentials on the local disk encrypted with a key derived from a master password. The second component is a browser extension that helps the user save passwords into the vault and to retrieve them for login pages. The extension provides an easy way for the user to log in to online services without leaving the browser window. On the other hand, the native app takes care of storing the passwords securely on the local disk. The app acts as an IPC server, and the browser extension communicates with it via IPC methods. As shown in Figure 2.2, the MitMa adversary tries to intercept IPC communication between the two software components. It does this from a different login session.

We analyzed several widely used password managers and other desktop software with a local client-server architecture. Table 2.1 summarizes our findings. The IPC methods that are vulnerable have a server process listening for connections from client processes. There are two main cases of such

vulnerable IPC methods: (1) network sockets, where the server listens on the loopback interface and a specific port number (e.g., 127.0.0.1:8888) and (2) Windows named pipes, where pipe instances are placed in a namespace that is accessible to all users including guests. In both cases, the server process binds to an endpoint identifier (a port number or pipe name) and waits for client communication. In such situations, the MitMa adversary can conduct client and server impersonation and even man-in-the-middle attacks as will be explained in detail below.

Client impersonation is possible if there is no client authentication or the client authentication is weak. This may be the case because no threats are perceived against IPC, or they are not taken seriously. For example, we found password managers where the desktop app and browser extension communicate over network sockets. To conduct client impersonation, the adversary needs to know the port number on which the app runs, which can be found either from the product documentation and source code or by running the `netstat` command. Client impersonation with named pipes works similarly, but the adversary needs to know the pipe name instead of the port number. The pipe name may be a well-known constant that can be found in the source code, or the adversary can enumerate the existing pipe names with the `get-childitem` command. Client impersonation may enable the adversary to query the user’s passwords from the vault.

Server impersonation is possible if the adversary takes over the address where the legitimate server should be listening for connections. Essentially, the adversary needs to be there first, before the legitimate server, so the adversary can reserve the port number or create the first instance and set the access permissions on the pipe. In some cases, the MitMa adversary can instruct the browser extension to collect and send web-form data as well as other DOM elements from web pages. While the passwords are securely stored in the password vault managed by the benign password-manager app, the adversary’s fake server may obtain any text typed by the user, including credentials and personal data. Thus, server impersonation
Inter-process Communication

Table 2.1. Summary of vulnerabilities in desktop applications (in 2018)

<table>
<thead>
<tr>
<th>Applications</th>
<th>OS</th>
<th>Channel</th>
<th>Attack(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password managers</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Roboform</td>
<td>M</td>
<td>Net. socket</td>
<td>Client imp.</td>
</tr>
<tr>
<td>Dashlane</td>
<td>M, W</td>
<td>Net. Socket</td>
<td>Server imp.</td>
</tr>
<tr>
<td>1Password</td>
<td>M</td>
<td>Net. socket</td>
<td>Server imp.</td>
</tr>
<tr>
<td>F-Secure Key</td>
<td>M,W</td>
<td>Net. socket</td>
<td>Client imp.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Server imp.</td>
</tr>
<tr>
<td>Password Boss</td>
<td>W</td>
<td>Named pipe</td>
<td>MitM</td>
</tr>
<tr>
<td>Sticky Password</td>
<td>M</td>
<td>Net. socket</td>
<td>Client imp.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Server imp.</td>
</tr>
<tr>
<td>Hardware tokens</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIDO U2F Key</td>
<td>W</td>
<td>USB</td>
<td>Unauth. access</td>
</tr>
<tr>
<td>DigiSign</td>
<td>M,W,L</td>
<td>Net. socket</td>
<td>Client imp.</td>
</tr>
<tr>
<td>Others</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blizzard</td>
<td>M,W</td>
<td>Net. socket</td>
<td>Client imp.</td>
</tr>
<tr>
<td>Transmission</td>
<td>M,W,L</td>
<td>Net. socket</td>
<td>Client imp.</td>
</tr>
<tr>
<td>Spotify</td>
<td>M,W,L</td>
<td>Net. socket</td>
<td>Client imp.</td>
</tr>
<tr>
<td>MySQL</td>
<td>W</td>
<td>Named pipe</td>
<td>MitM</td>
</tr>
<tr>
<td>Keybase</td>
<td>W</td>
<td>Named pipe</td>
<td>Server imp.</td>
</tr>
</tbody>
</table>

OS: macOS (M), Windows (W), Linux (L); Channel: Network socket (Net. Socket); Attack(s): Client impersonation (Client imp.), Server impersonation (Server imp.)

may lead to data theft if the browser extension sends data to the server, or integrity issues if the client trusts security-critical data from the fake server.

*Man in the Middle (MitM)* simply means that the adversary impersonates the server to the client and the client to the server and passes messages between the legitimate client and server. When named pipes are used, the adversary creates two instances of a pipe, one for the client impersonation and the other for the server impersonation. However, when network sockets are used as the IPC mechanism, the legitimate server and attacker cannot bind to the same port number on localhost. Fortunately for the adversary, many applications implement *port agility*, where secondary ports are chosen from a predefined list if the primary port is taken. Port agility enables the MitM attack because the adversary can receive client connections on the primary port and connect itself to a secondary port on the legitimate server. If there is no port agility, the adversary can still perform a MitM attack if it can *replay messages* by alternating between the client and server roles. The rate of the messages passing through the attacker will be slow, but we found practical attacks that only require a
small number of such role reversals.

Unauthorized access to USB Human Interface Devices (HID), such as hardware security tokens used for two-factor authentication, is another attack where the MitMa adversary could exploit the local communication. Although these devices are not IPC servers in the traditional sense, the communication between HID and a desktop application also occurs within one computer. We discovered that MitMa adversaries could access USB HIDs plugged in by other users in the Windows OS. By sending requests to the USD HID device at a high rate, the adversary creates exploits a race condition to confuse the deputy [61] and get a wrong document signed when the user interacts. Suppose the adversary has already obtained a password, e.g., from the impersonation or man-in-the-middle attacks mentioned in this section. In that case, the adversary could also spoof the second authentication factor with its unauthorized access to the hardware security tokens.

2.3 Vulnerabilities in cryptocurrency wallets

Publication II presents a security analysis of cryptocurrency desktop wallet applications with a local RPC interface. The wallet apps help users manage their cryptographic keys and use the keys for cryptocurrency transactions. We discovered that a MitMa adversary could impersonate the communication endpoints of RPC channels used by the wallet apps and potentially steal the keys and the cryptocurrency.

The wallet apps provide a command-line or graphical user interface for the users to manage their key pairs and a remote procedure call (RPC) interface for other applications to access the wallet functionality, such as querying the account balance or making transactions. The wallet apps are accessed with a JSON-RPC ² interface at an HTTP server that runs on a specific port number on the localhost, and the other applications connect to it locally or remotely as clients. Examples of such RPC client applications included web-browser extensions³, third-party wallet applications that do not want to implement the cryptocurrency protocol by themselves⁴, and cryptocurrency exchange web platforms⁵. While the open interface promotes an ecosystem of blockchain applications, it also increases the attack surface of the wallet apps. For instance, there have been reports about attacks where cryptocurrencies have been stolen from wallets by exploiting remotely accessible RPC where authentication was not correctly configured [115].

²https://www.jsonrpc.org/specification
³Metamask Ethereum client (https://metamask.io/)
⁴Bitcoin Armory (https://btcarmory.com/)
⁵Bisq (https://bisq.network/) and Peatio (https://www.peatio.com/)
Table 2.2. Summary of vulnerabilities in cryptocurrency wallets (in 2019)

<table>
<thead>
<tr>
<th>Auth.</th>
<th>Wallet app</th>
<th>RPC client</th>
<th>Attack(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No auth.</td>
<td>Geth Parity</td>
<td>Metamask, web3.py</td>
<td>Server &amp; client</td>
</tr>
<tr>
<td></td>
<td></td>
<td>web3.js, web3 java</td>
<td>impersonation</td>
</tr>
<tr>
<td>Basic access</td>
<td>Bitcoin Core</td>
<td>bitcoin-cli, Armory</td>
<td>Server &amp; client</td>
</tr>
<tr>
<td>auth.</td>
<td>Bitcoin Knots</td>
<td>bitcoind-rpc, python-bitcoinrpc</td>
<td>impersonation</td>
</tr>
<tr>
<td></td>
<td>Qtum Core</td>
<td>qtum-cli, qtumjs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dash Core</td>
<td>dash-cli, dashd-rpc</td>
<td></td>
</tr>
<tr>
<td>Digest access</td>
<td>Monero wallet</td>
<td>monero-python, monero-nodejs</td>
<td>Server</td>
</tr>
<tr>
<td>auth.</td>
<td></td>
<td></td>
<td>impersonation</td>
</tr>
</tbody>
</table>

The primary solution for protecting the RPC interface against such attacks has been to block remote access to the interface so that only local processes on the computer can access it. Also, wallet apps usually require password authentication when accessing the RPC interface. While these security mechanisms may help against remote attacks, we argue that local adversaries remain a threat. Similar to the IPC-based vulnerabilities explained in Section 2.2, we found in Publication II that a malicious process running locally in the background can impersonate the communication endpoints of the RPC channel and potentially steal funds from the wallets.

Thus, a MitMa adversary can perform client and server impersonation on the local RPC interface. Some wallet apps implement basic or digest access authentication mechanisms. Nevertheless, these mechanisms are of no help here because they are implemented such that only the server authenticates the client. Table 2.2 provides a summary of all the vulnerable wallet app and RPC client pairs found. We observed that the vulnerabilities discovered in one client often existed in other clients for the same wallet app or cryptocurrency.

2.4 Discussion

Our findings highlight the importance of the often-overlooked local adversary model where non-privileged background processes can intercept communication inside a multi-user computer. We demonstrate its seriousness with various examples of widely-used applications and compromises of critical data. We show that the vulnerabilities are common and that exploiting them is not difficult. We also found secure, well-designed applications that make use of local communication. Such positive examples provide guidelines for developers to use local communication securely.
We observed that software developers are ambivalent about the security of local communication. On the one hand, we found software products that treat local communication as a trusted environment and have no security mechanism. On the other hand, we encountered products that attempted to authenticate or encrypt the local communication, but rarely with the same prudence as seen in communication over physical networks.

Software developers can utilize existing OS mechanisms to detect attacks from a MitMa adversary. For example, named IPC endpoints can use OS APIs to confirm that the server and client processes are in the same login session or owned by the same user. However, it is important to ensure that both server and client do such checks independently. Unfortunately, browser extensions running inside sandboxed browsers and wallet applications running inside containers are deprived of OS-specific APIs and incapable of performing such checks on the server.

It is worth noting that some IPC mechanisms are immune to attacks from the MitMa adversary and should be used whenever possible. For example, IPC mechanisms such as anonymous pipes and socket pairs, where both the endpoints are created simultaneously by the same process, prevent an untrusted process from getting to the middle. CFMessagePort available on macOS is another IPC method that provides security by restricting the interactions to a single login session.

We recommend that the developers should consider all local communication as if it takes place remotely over the Internet and use the standard cryptographic solutions. IPC channels on the local computer could, in theory, be secured with cryptography because the OS can establish a secure channel between two processes. Authentication methods for communication over insecure channels have been studied widely and can also be applied to local communication. However, the difficulties related to local address naming — e.g., uniqueness and ownership of names — cannot be solved by cryptography.

Furthermore, security-conscious users can take some protective measures to ensure each computer is personal to one user only. Since the attacks are performed by leaving a malicious process running in the background, spatial and temporal separation of user sessions can be enforced. One way to do so is to limit the number of users, for example, by disabling guest accounts, remote access, and the RPC interface. As a long-term solution, operating system vendors can discontinue the support for guest users and for multiple simultaneous login sessions.

The findings from Publication I and II create awareness about the importance of local communication to isolate users from each other. While the discussed defensive solutions provide a quick fix, we hope our work inspires the development of better tools to avoid the entire class of problems in the future. Moreover, the attacks we presented are one way for a non-privileged process to circumvent isolation boundaries within the computer. As oper-
Inter-process Communication

...ating systems strive to isolate applications from each other, contributions from our work seem to provide valuable insight into application-isolation mechanisms’ design in subsequent research works [62, 100, 118].
3. Virtual Private Networks

This chapter summarizes the findings from Publication III, which analyzed the security of Virtual Private Network (VPN) communication. In particular, we analyzed the desktop client software of various VPN services and discovered several security vulnerabilities.

3.1 Background

Public networks are considered to be insecure and untrusted channels due to the presence of network adversaries who can tamper or eavesdrop on the communication. Virtual Private Network (VPN) is a widely used technology by large corporations and regular internet users as a protection against such adversaries. VPN connects the communication endpoints with encrypted tunnels using standardized protocols. The tunnels form a protected virtual link or network between the endpoints. VPNs can be broadly classified as corporate and commercial VPNs. Both types of VPNs have seen a surge in usage in the recent years [48, 47]. Corporate VPNs (also known as enterprise or business VPNs) are set up by organizations to connect their geographically distributed corporate networks and to provide remote access for employees to the corporate intranet. On the other hand, commercial VPNs (also known as personal or consumer VPNs) are subscription-based services available to regular Internet users to tunnel their Internet traffic via a service provider’s gateway server located somewhere in the cloud. With approximately 31% of all internet users relying on them on a regular basis [81], commercial VPNs are mainly used for a wide range of personal purposes such as accessing content from a different geographic location, securing sensitive online activities while on a public Wi-Fi network, and avoiding censorship and surveillance by local governments, employers, and access-network operators.

Both corporate and commercial VPNs connect distributed communication endpoints with encrypted tunnels using standardized VPN protocols such as IPsec. With strong encryption and integrity protection, VPN protocols
Virtual Private Networks

offer security guarantees against network adversaries. However, the protocols assume the endpoints to be configured securely. Misconfiguration of endpoints could severely undermine the security of VPN protocols as we demonstrated in Publication III. More specifically, we investigated how misconfiguration of commercial VPN clients could allow a network adversary to decrypt the traffic.

In both corporate and commercial VPNs, the server endpoint is configured by system administrators, who typically have training and knowledge for securely configuring the endpoint software. On the other hand, a significant difference lies in how the client or user endpoints are configured. In corporate VPNs, the system administrators also configure the employee devices with the required client software. Also, in corporate VPN setups, mostly a single VPN protocol is used for establishing the tunnel, and the client software is fine-tuned and configured to support just that protocol. In comparison, commercial VPNs require regular internet users to configure the client software by themselves. Furthermore, commercial VPN providers support several VPN protocols and servers, and provide instructions to the users on how to customize them according to their individual needs. Commercial VPN providers may have their own client software, which configures a VPN protocol library based on fixed and default values or based on user input. Also, the VPN provider may only operate the gateway, in which case, they provide instructions for configuring third-party client software, which may be part of the client operating system or downloaded from the internet. In any case, users who often lack technical or security knowledge and training are prone to misconfiguring the client software.

Although the security community has scrutinized VPN services, vulnerabilities and misconfiguration of the client-side software have been overlooked. In Publication III, we analyze client configurations of commercial VPNs and provide insights on common configuration mistakes and how to avoid them. We collected the configurations by following the user instructions provided by the VPN service providers to install the client software on Windows, macOS, and Linux. Our analysis revealed that these configurations could lead to vulnerabilities that allow a network adversary to intercept the VPN user’s traffic. Also, we found that, in some cases, the adversary will also be able to steal the victim user’s login credentials, giving the adversary free access to the user’s VPN service subscription. This chapter summarizes our findings.

3.2 Vulnerabilities in commercial VPN clients

We analyzed 30 popular VPN services that support standardized or other common VPN protocols: PPTP [59], SSTP [80], L2TP/IPsec [18, 110], Cisco
We also considered the fallback or automatic protocol selection strategy that was offered in some of the commercial VPN services. In this strategy, the client application will automatically select the protocol for the user by trying different protocols sequentially until it successfully creates a VPN connection. This is helpful if the firewall in the access network blocks some VPN protocols or if the user does not want to understand the technical intricacies of choosing the right protocol.

Our analysis involved two steps. Firstly, for each of the selected services, we analyzed the VPN clients with the provided configuration instructions and unchanged default settings. We looked for potential misconfigurations that might compromise the security of the VPN connection. Secondly, if a potential misconfiguration was found, we analyzed the resulting vulnerability and, when feasible, implemented an exploit. As summarized in Table 3.1, our analysis revealed several vulnerabilities that a network adversary could exploit. Among the 30 VPN services, every service had at least two vulnerabilities. While Publication III reports each vulnerability in detail, this chapter categorizes the vulnerabilities into four types and presents a summary with examples as follows.

**Unclear instructions:** We found that many commercial VPN services provide instructions for configuring VPN clients that are unclear, incomplete, or insecure. For instance, the built-in VPN client in Windows 10 does not enforce encryption by default on any PPTP connections. We found that 14 out of 21 VPN services that support PPTP protocol do not instruct their users to change this setting while configuring PPTP with the built-in client on Windows 10. A network adversary can misuse this to negotiate with the client not to encrypt its traffic, and the client agrees to do so because it is not mandatory to use encryption. Then, the network adversary can obtain all traffic as if no VPN was used.

**Faulty implementation:** The second type of vulnerability is due to faulty implementations of third-party software components used by VPN clients. Typically, these components are software libraries integrated into the OS or installed by the users. We found a faulty implementation in sstp-client\(^1\), a library used in Ubuntu for creating VPN connections based on Secure Socket Tunneling Protocol (SSTP). SSTP uses HTTPS to create an encrypted tunnel between the endpoints. While setting up the tunnel, the client opens an HTTPS connection to the server and verifies the server's certificate as in any HTTPS connection. The sstp-client library on Ubuntu allows the user to configure whether the connection should be terminated when the server certificate verification fails. However, by inspecting the source code, we discovered that sstp-client always ignored certificate verification errors and established the VPN connection irrespective of the

\(^1\)https://sourceforge.net/projects/sstp-client
Virtual Private Networks

Table 3.1. Summary of VPN client vulnerabilities (in 2019)

<table>
<thead>
<tr>
<th>Protocol</th>
<th>OS</th>
<th>Vuln. clients</th>
<th>Vulnerability description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPTP</td>
<td>W</td>
<td>14 /21</td>
<td>Encryption is not enforced by the OS</td>
</tr>
<tr>
<td>SSTP</td>
<td>U</td>
<td>4 /4</td>
<td>Ignored certificate verification failures</td>
</tr>
<tr>
<td>L2TP/IPSec</td>
<td>W, M, U</td>
<td>26 /27</td>
<td>Known pre-shared keys to setup tunnels</td>
</tr>
<tr>
<td>Cisco/IPSec</td>
<td>W, M, U</td>
<td>14 /14</td>
<td>Known pre-shared keys to setup tunnels</td>
</tr>
<tr>
<td>IKEV2</td>
<td>U</td>
<td>3 /5</td>
<td>Client accepts any certified server certificate regardless of its identity</td>
</tr>
<tr>
<td>SoftEther</td>
<td>W, M</td>
<td>8 /8</td>
<td>Server certificate not verified</td>
</tr>
<tr>
<td>Fallback</td>
<td>W, M</td>
<td>4 /9</td>
<td>Use of weak protocols as fallback options</td>
</tr>
</tbody>
</table>

OS: macOS (M), Windows (W), Linux (L); Vuln. clients: No. of vulnerable / No. of supported

user’s configuration. Ignoring the certificate verification failure allowed a network adversary to impersonate a legitimate VPN server and obtain all the victim’s traffic. All four VPN services that support SSTP were vulnerable to this attack.

**Insecure default configuration:** Most end-users rely on the default configuration of the software they are using. Hence, the configuration should be secure by default and not require users to adjust it. Unfortunately, many commercial VPN services fail to ship with a secure default client configuration. One such case is the SoftEther VPN clients for Windows and macOS. SoftEther is an HTTPS-based protocol in which the verification of server certificates is a critical security requirement. SoftEther VPN connections have a parameter called `CheckServerCert`, which is set to `False` by default. We found that all of the eight VPN providers that support SoftEther retained this insecure default setting, due to which a client will not verify the server certificate. Consequently, a network adversary can perform server impersonation on the VPN connection and obtain all the network traffic of the client.

**Failure to disable legacy protocols:** As a rule of thumb, legacy protocols or components with known security issues should be disabled. PPTP, with its known cryptographic weaknesses [83, 63], is one such protocol. Among the 30 commercial VPN services in our study, 21 of the VPN services supported PPTP. Another protocol that is still widely used despite known weaknesses is `L2TP/IPSec`. It uses publicly known pre-shared...
Virtual Private Networks

keys that an adversary can easily obtain and perform MitM attacks. We found that 26 out of the 30 commercial VPN services in our study still had this vulnerability. Furthermore, PPTP and L2TP/IPsec should be strictly disabled in automatic protocol negotiation or a fallback strategy because the attacker can choose the weakest protocol that the client supports. We found that, 4 out of 9 VPN services that offered the fallback strategy that implemented automatic protocol negotiation had used PPTP or L2TP/IPSec as one of the options.

We also found positive examples in the case of OpenVPN. Despite the wide range of configuration options that OpenVPN supports, we did not find broken configuration examples that would allow the network adversary to compromise the VPN tunnel. Nevertheless, we discovered that several OpenVPN clients are susceptible to the MitMa local adversary. The reader is referred to Chapter 2 and Publication III for more details on the security issues that local adversaries can cause in VPN clients.

3.3 Discussion

It appears that commercial VPN services compete by providing the maximum number of features, such as different VPN protocols, and that security is a secondary concern for them. We found many unnecessary flaws in the VPN client settings, such as not checking the server certificate or name. Our findings show that security flaws are not always deeply hidden in the code or cryptography. Instead, simple configuration mistakes, poor instructions, insecure default values, and failure to disable legacy features can also result in widespread security failures across an entire industry. While each of the vulnerabilities alone might seem like a trivial mistake, together they indicate a lack of technical knowledge across the commercial VPN service providers. We hope our work increases awareness, at least to some extent, about the importance of correct configuration among VPN service providers and software developers.
4. **Cellular Communication**

In this chapter, we summarize the findings from Publications IV, V and VI, which analyzed the security of cellular communication based on 3GPP Standards. Publications IV and V present two specific attacks that exploit signaling protocols used in the backend networks between mobile operators. Publication VI presents the *Bhadra framework*, a domain-specific conceptual framework for modeling threats and attacks against cellular mobile networks.

4.1 **Background**

Mobile communication systems comprise various subsystems and interconnected networks. The Radio Access Network (RAN) forms the first leg of mobile network communication, where the user devices connect to a Mobile Network Operator (MNO) over wireless channels. Then, the RAN connects to the Core Network (CN), which is the backend of the MNO. The CN is responsible for managing the mobility of the users, initiating connections with other MNOs, and delivering telecommunications services, such as voice calls, SMS, and data connections. In addition, the CN is connected to other subsystems that are responsible, e.g., for billing and charging and value-added services. Step changes in cellular mobile communications network technology and architecture, called *generations*, are standardized by the 3rd Generation Partnership Project (3GPP). Each generation is expected to overcome the limitations of the previous generation with improved capabilities, such as bandwidth, latency, and security. We refer the reader to Rost et al. [98] for more details about the evolution of the mobile network architecture. Our work focuses on the currently co-existing generations of 3GPP cellular communications from 2G to 4G. While Publication IV and V focus on the backend core networks, Publication VI considers the entire mobile network.

The majority of the communication within the core network is carried out by signaling protocols that control various network components and
functionalities. The 2G and 3G generations of mobile communications networks use Signalling System 7 (SS7) or SIGTRAN, the adaptation of SS7 over IP, as the signaling protocol. The SS7 protocol suite was developed when mobile phones did not yet exist. SS7 was designed to establish reliable connections between the wired Public Switched Telephone Networks (PSTN) of mutually trusting state-owned telecommunication operators. Since the telecommunications networks were accessible only to trusted partners, the protocol design did not consider strict access control policies and cryptographic building blocks. Eventually, the telecommunication networks evolved and replaced wired phones and networks with mobile phones and cellular networks. However, the telecommunication operators’ networks are still connected to each other via SS7 interconnections. Furthermore, modern-day global mobile communication networks consist of many private entities, such as service and infrastructure providers, who could have access to the SS7 network but often lie outside the operators’ control. Such entities could misuse their access and exploit the lack of security in the signaling protocols and interconnection networks. Unfortunately, exploiting signaling protocols has also become a part of surveillance and cyber-espionage practices [76].

SS7 is predominantly used for international roaming, where local operators of that country will serve a mobile subscriber roaming abroad. The visited network communicates with the roaming subscriber’s home operator’s network using SS7. However, since the SS7 protocol does not have a mutual authentication mechanism, a potential adversary can spoof signaling messages from a roaming partner. Dishonest or compromised MNOs and law enforcement agencies who misuse their access and anyone who could purchase SS7 access through illegal marketplaces [27] could be potential adversaries. The adversary might also benefit from having some knowledge of the user-specific identifiers and core network infrastructure and protocols. For instance, an adversary with access to the SS7 network and knowledge of the subscriber’s identity (IMSI or phone numbers) can impersonate a benign visited network to the subscriber’s home network by claiming that it is serving a roaming subscriber.

Security literature presents attacks where core network adversaries can potentially learn the precise location of a mobile subscriber [39, 41], eavesdrop on its voice calls, intercept its SMS [86, 40, 66] and commit financial fraud [94, 73]. We extend this research theme in Publications IV and V. We study how an adversary with core network access could misuse the signaling protocols. Mobile networks also suffer from attacks by adversaries other than those with access to core networks, and Publication VI studies them. We present a threat modeling framework that provides a structured way to describe and characterize known threats and attacks against 3GPP-based mobile networks. The framework can be used to analyze and communicate threats on a level that abstracts away the
technical details but still provides meaningful insights into the adversarial behavior.

4.2 Vulnerabilities in mobile network signaling

It is important to note that not all security vulnerabilities are bugs. Instead, a vulnerability could be an intended behavior that exposes the system or its users to a security risk. Such risks arise due to evolved threat landscape, such as advancements in adversarial capabilities, adding new features, or involving more actors. Mobile communications systems, especially the legacy signaling components or subsystems, are a typical example of how the intended behaviors of the past become the security vulnerabilities of today. Most SS7-based attacks are about finding misuse cases of a legitimate functionality, where an adversary uses the functionality in an unexpected and malicious way. Publication IV discusses one misuse case in detail.

The Equipment Identity Register (EIR) stores International Mobile Equipment Identities (IMEI), unique identifiers provisioned by mobile device manufacturers. Typically, the EIR maintains a denylist of IMEI of devices that have been reported as stolen and blocks the devices from using the mobile network. Every time a mobile device is switched on or moved to a new location, the IMEI is sent to the mobile network during the registration process as part of the CheckIMEI standard procedure [8]. The network access is granted only if the EIR confirms that the device is not on the denylist. The 3GPP standards also allow operators and manufacturers to perform advanced checks by including additional parameters in the checking procedure. For example, various networks use the International Mobile Subscriber Identity (IMSI) in a variant of the checking procedure called Enhanced CheckIMEI. Here, the EIR keeps a list of IMEI-IMSI pairs instead of just IMEIs. During mobile device registration to the network, if the IMEI is found in the denylist, an additional check is made to verify whether the IMEI-IMSI pair exists in the EIR, and the device will be automatically moved out of the denylist if found. This feature is supported by EIR to automatically unblock lost mobile devices if they return to the network with the old IMSI. The operator reasons that the owner has regained the possession of the device. However, as we explain in Publication IV, this feature can be misused by an adversary with access to the SS7 core network. The EIR deployments are currently specific to the operator and country, and each operator has to maintain their own denylists independently. However, there is a proposal to centralize the EIR so that denylisted devices could be barred globally from using any mobile networks [50]. The centralization would enable an adversary with access to the SS7 network to unblock any stolen device in the world. Thus, this
potential vulnerability needs to be addressed before progressing with the centralization.

There could be other ways to misuse SS7 features that are yet to be uncovered. Thus, it is important to understand the implications of SS7-based attacks on signaling protocols developed after SS7. In this realm, **Publication V** provides a basis for studying the security of the transition from SS7 to its successor, Diameter. In this work, we investigated whether Diameter inherits security problems similar to SS7 and the possibility of translating SS7-based attacks to Diameter-based attacks.

At first look, the Diameter protocol appears to offer advanced security features that were missing from SS7. 3GPP TS 32.210 specifies the use of Diameter with IPsec for securing the signaling [10]. However, there is a significant gap between the standardization and actual implementation. Most of these features are not mandatory, and many operators have not deployed them. Furthermore, in the global roaming network of signaling interconnections between operators from different countries, there is no consensus on how to deal with the management overhead of Diameter. At the time of writing Publication V, the discussions around how and who manages and pays for the public key infrastructure were still ongoing. Managing such global infrastructure always involves a political dimension, which slows down the technical development.

In our research, we found that the interconnection where the co-existence of SS7 and Diameter creates an interesting attack surface that had never been studied before. The SS7 to Diameter upgrade is a gradual process that requires significant changes to the underlying infrastructure to avoid service interruptions. So, the operators require signaling between the SS7-based 2G and 3G core networks and the Diameter-based 4G core network. Such inter-generation and inter-protocol communication is standardized as Interworking Function (IWF) in 3GPP TS 29.305 [9] and TR 29.805 [7].

Publication V investigates how a backend adversary can undermine the additional security features of Diameter by misusing the IWF. We found that an SS7 adversary can carry out the attacks without extensive knowledge of the IWF, Diameter, and 4G networks. Interestingly, the IWF

---

**Figure 4.1.** Interworking between 2G/3G and 4G core networks
allows an adversary from the less secure, 2G and 3G networks to attack 4G and newer networks. As shown in Figure 4.1, the edge nodes of the SS7 and Diameter core networks — Signal Transfer Point (STP) and Diameter Edge Agent (DEA), respectively — are configured to automatically translate SS7 and Diameter messages with the IWF. Unfortunately, no additional security checks are done at these edge agents to filter out attack traffic that IWF messages can carry; this is an ideal situation for the adversary to launch attacks. We found in Publication V that two out of five SS7-based location tracking attacks can be easily converted into Diameter attacks without additional effort [96].

4.3 Bhadra — a domain-specific threat modeling framework

After over three decades of evolution, mobile networks have become a complex ecosystem of business, components, and interfaces that combines communications and security technologies and trust models from different eras. The mobile networks constantly evolve to support the growing need for faster and more reliable communication while maintaining interoperability between the industry players and backward compatibility with the previous technology generations. Due to their complexity and scale, it is difficult to form a comprehensive picture of the security architecture and the threats against mobile networks. We address this gap in Publication VI with the Bhadra framework, a conceptual framework for modeling threats and potential attacks against mobile networks.

The Bhadra threat modeling framework focuses on the mobile communication domain. Within that scope, Bhadra is sufficiently agnostic about the underlying technologies to model domain-specific threats against both the current and future generations of the technology. It aims to provide a structured way to analyze and communicate threats on a level that abstracts away the technical details but still provides meaningful insights into adversarial behavior. Our work is inspired by MITRE ATT&CK, a popular framework for modeling threats against enterprise information systems [102]. Although the ATT&CK framework is used even in the mobile network industry to model threats against the enterprise components, the framework focuses mainly on the platform- and OS-level threats. Moreover, it also excludes all the threats that are exclusive to the mobile network. Therefore, we complement ATT&CK with the Bhadra framework by focusing specifically on threats and attacks against widely deployed 2G, 3G, and 4G networks.

The Bhadra framework studies attacks on mobile communications from different types of adversaries (refer to Figure 4.2). We surveyed the known attacks against mobile communication networks from two types of literature. Firstly, we collected peer-reviewed academic and professional
security research publications that analyze independent subsystems and components of mobile communication, such as the mobile device [51, 58], radio access [25, 99] and core network [39, 77]. Secondly, we gathered public resources provided by standardization bodies (e.g., 3GPP, GSMA, ETSI) and government agencies (e.g., ENISA and NIST). These resources describe classes of threats, as well as best practice guidelines for defensive strategies recommended by industry experts. Then, we organized all the attacks and potential threats into attack phases, tactical objectives, and techniques. The Bhadra framework categorizes publicly known attacks into nine tactical categories and 55 different techniques. We use concrete case studies to show how the framework captures adversarial behavior. Publication VI can also serve as a survey of publicly known threats against mobile telecommunication.

4.4 Discussion

Mobile operators were aware of the signaling systems’ security issues from the beginning. However, the general security research community became aware of issues after the public disclosure of the SS7 attacks in the year 2008 [39]. Since then, news media has publicly discussed the increase and prevalence of the SS7 attacks in tracking the locations of politicians [26, 89], stealing money from bank accounts [33], stealing SMS-based two-factor authentication tokens [34, 46], and surveillance campaigns with political intentions [24, 71, 72, 76, 23]. The public knowledge of the signaling attacks and their impact have also led to legal and political discussion [111, 114].

One of the characteristics of signaling attacks is that it is challenging to differentiate attacks from benign traffic. Any SS7-based communication,
including attacks, is part of legitimate mechanisms, such as location updates, voice call setup, SMS delivery, and emergency call. For this reason, firewalls may not be able to efficiently filter signaling attacks. Although a few firewalls have been advertised as specifically designed to filter signaling attacks [68, 101], there is no public information about their efficiency or their deployment in the wild. Another characteristic of signaling attacks is that mobile end users have no means to know whether they are under attack. The Sonar distance bounding mechanism by Peeters et al. is a rare solution that can detect some of the signaling attacks from the mobile user’s side [93]. Previous research has disregarded peer-to-peer signaling attack detection mechanisms, which can be carried out without the operator’s intervention. In this regard, solutions like Sonar open up new directions for research and security software development.

Generally, security failures are not discussed openly or between business associates in the mobile communication industry. One reason is that there was no easy way to share information about threats and potential attacks. The Bhadra framework addresses this issue by providing a conceptual framework, abstractions, and tools to communicate information to engineers, managers, industry partners, and customers. On the one hand, Bhadra provides a structured way and common terminology for discussing the attacker’s behavior. The level of abstraction provided by Bhadra aims to share meaningful information without disclosing confidential details about the operator’s network. Thus, Bhadra is useful for threat intelligence sharing across organizations. On the other hand, Bhadra is a knowledge base of known attacker tactics and techniques. The knowledge base is systematized following the attack phases from target discovery all the way to impact. Since the framework is structured to model the attacker’s behavior, it can also help in attack attribution.

The Bhadra framework has so far been used for company-internal threat modeling by Nokia Bell Labs. We have also implemented threat modeling tools for internal use [31]. Furthermore, the Bhadra framework has already influenced an industry-wide effort [37]. Kicked off by the foundation work provided by Bhadra, Formal work on a threat modeling framework is now in its starting phases in GSMA under the banner of Mobile Threat Intelligence Framework (MOTIF) and in MITRE under the FiGHT project [107]. We will make the Bhadra knowledge base available to the public through contributions to these new community efforts.
This chapter summarizes the findings from Publications VII and VIII, which analyzed the security of API-based communication. Publication VII presents Cross-site Scripting (XSS) attacks against cloud applications and their add-ons through an empirical analysis. Publication VII presents a qualitative study with software developers that uncovered usability and security pitfalls of Trusted Platform Module (TPM) library APIs.

5.1 Background

Modern software development relies on Application Programming Interface (API) for establishing communication between software components. Technical details on how to connect the components using APIs are described in product-specific documentation or in a standard specification. We studied two different types of APIs.

In Publication VII, we analyzed the APIs for cloud application microservices. Most cloud applications follow the microservices architecture, where the functionality is implemented as multiple independent services which connect to each other via APIs. The cloud applications may also provide APIs for extending their core functionality and implementing new features. The additional features implemented on top of the core services are referred to as add-ons. The ecosystem of add-ons around cloud applications is a relatively new phenomenon, and its security has not been widely studied. Publication VII studies the effects of add-ons on cloud application security. Section 5.2 provides a summary of our findings.

In Publication VIII, we analyzed the APIs for hardware interfaces of Trusted Platform Module (TPM). These are the high-level APIs that allow software applications to communicate and access TPM’s core capabilities [104, 105, 106]. We studied the implementation of the TPM APIs by various software libraries. Our research methodology combined usability and human factors analysis with technical analysis. The motivation to do so comes from our observation in Publications I, II, and VII that there could
be several instances of human errors that could undermine the security of software applications. On the one hand, we found many instances of security flaws due to poor instructions and documentation of software libraries, the use of insecure default values and legacy features, and configuration mistakes. On the other hand, we found that developers lack security awareness, and they underestimate adversarial capabilities or cannot fully evaluate emerging threats. So, we sought inspiration for analyzing the TPM library APIs from previous work that explored usability and human factors of security [90, 12, 55, 60, 17, 85, 11]. Section 5.3 provides a summary of our findings.

5.2 Vulnerabilities in cloud applications add-ons

Cloud application add-ons add customized commands and features to their host application. Each add-on is a separate microservice with its own server and client components. In addition, the add-ons generally have a web front-end that is embedded into the host application's user interface. The add-ons access user data and some core functionality of their host through APIs. Cloud applications typically use permission-based access control and OAuth access tokens to enable their access to user data. Each add-on has a list of permissions that it requires to operate. The user must explicitly review and approve the permissions when the add-on is installed or used for the first time. If the add-on is updated and requires new permissions, the user must review and approve them again. The add-on UI can interact with the host application in two ways: (1) locally inside the browser, where the add-on UI interacts directly with the host application UI, and (2) in the backend via the add-on server in the cloud, which can interact with the host application server.

Successful cloud applications have also created marketplaces for add-ons that have attracted third-party software developers. On the one hand, many add-ons in these marketplaces are quick hacks by inexperienced developers who may not fully understand security risks of add-ons. On the other hand, the cloud application vendors have an incentive to attract new add-on developers to their ecosystems, which may lead to less stringent security controls for the add-ons than for the core service. Hence, although the add-ons provide additional services on top of the core services of the cloud application, they can also introduce security issues.

In Publication VII, our analysis revealed that many add-ons are vulnerable to cross-site scripting (XSS), a widely known client-side code injection attack [57]. We discovered two ways in which an adversary can conduct XSS attacks on cloud applications. The first approach involves client-side code injection via a shared workspace (refer to Figure 5.1a). Here, the attacker and the victim use the same cloud application and share a
workspace, i.e., a collaborative editing environment where changes made by one user are propagated to the other users. The attacker injects malicious JavaScript code into the shared workspace and hides it in the plain sight of the victim, for example, by changing the font color to match the background or by using tiny fonts. When the victim enables the vulnerable add-on for the shared workspace and the add-on renders the attacker's input in an unsafe way, the injected script may become part of the web page in the add-on iframe and be executed by the victim's web browser. In the second variant (Figure 5.1b), the malicious code injection occurs with outside input in cloud applications that accept external input such as email messages from non-users. The rest of the details of hiding the input and code executions remain the same.

Irrespective of how the attacker injects the code, the root cause of the attacks is that the vulnerable add-ons do not process untrusted user input in a safe way. The injected code may be able to steal user data in two ways. First, since the malicious code runs in the add-on iframe, the adversary could call the APIs exposed by the add-on server and access the user data stored there. Second, the adversary can also use HTML5 APIs on the browser to request access to resources on the victim's local machine. It is important to note that the malicious script runs in the add-on iframe with a different origin than the host application. Hence, the same-origin policy restricts the script from directly accessing any data in the host application.

**Architecture analysis:** We analyzed the host-application architecture to understand what an adversary could gain from XSS attacks. We selected 3 popular cloud-application suites: Microsoft (MS) Office Online\(^1\), G Suite\(^2\), and Shopify\(^3\). As shown in Figure 5.2, each application suite had a different architecture. Our analysis revealed three types of XSS exploits.

\(^1\)https://office.com/
\(^2\)https://gsuite.google.com/
\(^3\)https://shopify.com/
Firstly, the adversary was able to get the same level of access as the add-on. Because of the local messaging with the host application window in Microsoft Office Online, the adversary was able to access any resources that the add-on was permitted to access. In G Suite, the applications did not accept local messages from the add-ons. However, this limitation could be bypassed by using the Picker API to gain the same permissions to the user’s data as the add-on server.

Secondly, we found that an adversary was able to create a malicious add-on or app with the same name as the vulnerable add-on and display an authorization prompt to request an OAuth 2.0 token. Suppose the victim authorizes the adversary’s application; the adversary would have received an access token that gives access to the host application via APIs that are far more powerful than the client-side add-on APIs. Both Microsoft Office Online and G Suite were susceptible to this type of exploit.

Thirdly, we found that an adversary was able to trick authorized users

\footnote{https://developers.google.com/picker/}
into installing malicious add-ons in Shopify. Shopify allowed users to initiate the installation process by visiting a URL. Thus, the adversary had to create a malicious add-on with a similar name as the vulnerable add-on and initiate the installation from a URL endpoint that the adversary controls. The victim would be tricked into thinking that the add-on has been updated and must be authorized again.

**Empirical analysis:** To find out how common the XSS vulnerability is in cloud application add-ons in the wild, we conducted an empirical analysis by looking for vulnerable add-ons in the marketplaces of the three selected cloud application suites. Our focus was on non-malicious add-ons written by well-meaning developers who do not intend to cause harm but might not be security experts. We selected 50 popular and 50 random add-ons from each of the three marketplaces. Then, we manually analyzed all 300 add-ons and found that 28 of them (9%) are vulnerable to the XSS exploits described in this chapter. We observed that the vulnerability rate in the set of popular add-ons is lower in all three marketplaces. We speculate this could be because popular add-ons were written by experienced developers. Also, we observed that add-ons that are vulnerable to outside input are rare, and we found only one such example. This could be because the add-on developers are more familiar with threats from external inputs, such as emails, than those from a shared workspace. Nevertheless, our results indicate that XSS vulnerabilities are common in cloud application add-ons, and exploiting them is not difficult.

### 5.3 Usability of trusted platform module library APIs

A Trusted Platform Module (TPM) [103] is a tamper-resistant chip that is used as a hardware-based root of trust in many modern applications [64, 113]. TPMs can carry out common cryptographic operations, such as secure key generation, encryption, hashing, and signing. Furthermore, since the TPM is physically isolated from the processing system of its host, it can be used for securely storing a small amount of sensitive data (e.g., keys and certificates) that can verify the integrity of the host. TPMs also provide various non-cryptographic security features for imposing access control restrictions on the objects created or stored in the TPM. Such restrictions play a crucial role in hardening the security of software applications built using TPMs. The Trusted Computing Group (TCG) defines standard specifications that cover TPM architecture and implementation [109]. In addition, the TCG has also defined several high-level APIs to interact with the TPM hardware and utilize its functionalities [104, 105, 106]. These high-level APIs are implemented by various software libraries and is the object of our study in Publication VIII.

We analyzed tpm2-tools, a widely used TPM library API. The main goals
of our study were to understand the usability and security pitfalls of TPM developers and to review the current API implementation to provide concrete design guidelines for usably secure API development. We conducted a qualitative study using task analysis, survey questionnaires and semi-structured cognitive interviews with TPM developers. To this end, we implemented a study environment which is now released as open source to support further studies. We designed four simple tasks around common use cases of TPM to understand whether a knowledgeable developer could choose suitable security parameters. The tasks required the participant to use well-known TPM commands. The questionnaires and interviews attempted to understand the participants’ experiences and perceptions about the APIs.

Our analysis included two phases. In the first phase, we analyzed the code snippets collected from the study environment for correctness and
API-based Communication

Also, we tried to understand the typical solutions and coding patterns that affected the security of the participants’ code. We found that the participants rely heavily on the default values of security attributes — such as cryptographic algorithms and key length — when provided by the library. The participants were careless or made insecure choices when deciding the attributes themselves. This pattern emphasizes that software libraries should provide secure defaults, and the documentation should include correct and secure code examples and guidelines for picking the security attributes. We found that developers do not always feel forced to make security choices unless a threat model is given or they understand the threats to defend against. Documentation could help the developers by including common threat models to cultivate intuitive security thinking and form correct mental models. In the second phase, we conducted a thematic analysis of the interview transcripts. We identified three categories of usability and security issues as summarized in Table 5.1. The library theme captured the usability issues of the library software. The supporting materials theme captured the shortcomings of library documentation. The user theme depicted the factors influencing the developers’ choices and mental models.

5.4 Discussion

Add-ons in cloud applications are a relatively new phenomenon whose vulnerabilities have not been widely studied. In Publication VII, we analyzed the security of API-based communication used in cloud applications and their add-ons. We found vulnerable add-ons in the wild and showed that these vulnerabilities might introduce new security threats to their host applications. In particular, the add-ons do not always take care when processing untrusted input, which can make them vulnerable to XSS attacks. Since XSS is a well-known vulnerability, prudent engineering practices have been developed to prevent such mistakes [91]. On the one hand, developers are aware of the need to filter untrusted input, and on the other, cloud application vendors have developed platforms and toolchains that make their products immune to most types of code injection. Nevertheless, the problem has not been completely solved. Attackers always find new ways of bypassing defenses, as we show in Publication VII. Moreover, the speed of software development makes it difficult for threat analysis and defenses to stay up to date. Our work does its part to catch up with the development in one key area of modern software, i.e., add-on ecosystems.

Publication VIII demonstrates that tpm2-tools APIs, as designed, are not developer-friendly. Although various guidelines are available to design usably secure APIs, we find that the tpm2-tools library has not seriously considered or implemented them. Consequently, developers struggle to
use the APIs efficiently and are prone to make trivial mistakes that undermine security. The complexity of the topics and lack of supporting materials could pose a major barrier for developers. We provided explicit recommendations, in the form of concrete action points, for improving the documentation and software. We also found that some usability and security pitfalls can be traced back to the TPM standard specification that forms the design basis for the software implementation. Based on this observation, we highlight an important issue: any technology that follows standard specifications tends to accumulate usability pitfalls already during the standard specification. This is an opportunity for standardization bodies to prioritize usability by involving HCI experts in the design and review of standards to preempt any possible usability pitfalls that otherwise could be propagated to the software implementation.
6. Lessons Learned

In this chapter, we present common observations across the publications and discuss potential future research directions.

6.1 Need for a standard channel for vulnerability reporting

During our research, we analyzed a large number of software products and services. The disclosure of the vulnerabilities to the vendors took as much work as finding them. The experience varied from one vendor to another. In some cases, vulnerability reporting was easy, and the vendors quickly fixed them. However, there were several occasions on which we were unsure whom to contact, had no means of reaching out, or did not receive any acknowledgment or updates on the security issues that we reported. Of all the processes for reporting vulnerabilities, we believe the channel for communicating with the vendors plays a crucial role. Therefore, we now present our experiences, grouped by the communication channel, and share our opinions.

Bug bounty programs: Many of the software products and services we analyzed had bug bounty programs, such as HackerOne and BugCrowd, as the only channel for reporting security flaws. Bug bounty programs are a cost-effective method of crowdsourcing security efforts, where any individual can report security issues and earn financial rewards [112]. However, such programs may not be ideal for identifying new types of security issues that are not familiar to the community. Most bug bounty reporters reproduce known security issues, e.g., from OWASP top 10, and use security automation tools such as Metasploit to earn more bounties quickly and with less effort. Therefore, it is unlikely that the newer types of security flaws that use novel techniques and threat actors will take precedence over a large number of already-known types of security flaws submitted through bug bounty programs. Furthermore, given the monetary rewards, the bug bounty programs are naturally skewed towards serving those who find large numbers of easy-to-verify vulnerabilities.
Lessons Learned

From our experience of using bug bounty programs for responsible disclosure of research results, we found them to be problematic for two reasons. Firstly, the bug bounty programs required us to associate the discovery with a weakness derived from MITRE’s Common Weakness Enumeration (CWE)\(^1\), a list of common software and hardware weakness types. In general, associating a newly discovered security flaw with CWE provides a common language and well-established terminology for communication between bug reporters and vendors. However, many security weaknesses discussed in this thesis required us to convey a new type of adversary or adversary model. By associating with CWE, in some way, we had to restrict the scope of the reports to already known types of security weaknesses and only report individual software bugs rather than discussing the impact of the new adversary models. We believe new types of threats are more crucial from the security research point of view.

Secondly, we found the monetary rewards to be problematic for academic security researchers. These rewards may encourage ethical reporting of security weaknesses. However, some bug bounty platforms forbid publishing any information about the vulnerabilities even after they are patched. This practice hinders academic security research, where the actual incentive lies in disseminating the knowledge to the larger community.

**Customer support or product help desks:** For some products and services, we had to rely on customer support or product helpdesks to report the vulnerabilities. We contacted them through a wide variety of media, such as human-assisted chatbots, social media platforms like Twitter and Facebook, emails to customer support, and online ticketing systems. While most of these media might be suitable for making a product- or service-related query, they are not really suitable for communicating security-related issues. Customer support and help desk personnel are typically not equipped with the required security knowledge to understand the consequences of security vulnerabilities. They focused on solving the immediate problem for the customer rather than solving the security issue in the product. Furthermore, they did not have the sense of urgency that is required for vulnerability reporting. We found that customer support helpdesks rarely prioritize security-related issues and often treat them as regular customer feedback. Moreover, we found that queries made through help desks are sometimes shared on product community forums. Security weaknesses posted in the product forums could also be read by adversaries.

**Email:** We reported many vulnerabilities by sending email to the respective software companies. However, not all companies had dedicated security contact addresses. In these cases, we emailed their data protection officers, who sometimes handle security breaches. Their contact details are available publicly for companies in the European region. We

\(^1\)https://cwe.mitre.org/
also copied the reports to other available contacts who may have a role in security decisions, such as the chief technology officer. In some cases, we had to send unsolicited email messages to generic addresses such as contact@companyname.com. Not having a dedicated email address to report security-critical information may discourage responsible disclosure.

Creating an issue in a public issue tracker: We found some of the security weaknesses in widely-used open-source software libraries. These libraries are handled by volunteer communities that are not sufficiently funded to run bug bounty programs or have dedicated support staff to resolve security issues. Therefore, software bugs, including security vulnerabilities, are reported as an “issue” on the public code repositories. While the issue trackers distribute the workload of bug fixing among the volunteer community, they are not suitable for reporting security vulnerabilities. Similar to public product support forums, reporting security vulnerabilities publicly as issues on code repositories could lead to exploits before the problem is fixed.

Reporting with organizational affiliation: At the time of publication of Publication IV and V, the only way to report our findings to mobile operators was by participating in GSMA working group meetings. Unfortunately, this participation was open only to member organizations of GSMA and not to individual researchers. Hence, one of the co-authors with industry affiliation had to present the findings about potential misuse of signaling protocols. Recently, GSMA has initiated an online Coordinated Vulnerability Disclosure (CVD) program\(^2\), where anyone can report a security issue. We believe the CVD program to be an efficient and transparent tool for responsible disclosure.

Based on these experiences, we believe there is a need for a standard channel for reporting security issues. Also, creating easy ways to report vulnerabilities could positively impact responsible disclosure. One proposal is in the Internet draft security.txt\(^4\). It unifies all the necessary information to report security issues in one place and provides instructions for those who want to conduct responsible disclosure. The idea is to place a text file called “security.txt”—with contact information, PGP keys, channels, and procedures to report security issues—in a well-known location\(^8\). It works similarly to robots.txt but is intended for humans wishing to report security issues. While many companies have already adopted the proposal, standardization and regulation could increase its acceptance.

\(^2\)https://www.gsma.com/security/gsma-coordinated-vulnerability-disclosure-programme/
6.2 Tackling software dependencies

Modern software development depends on various third-party components, such as programming libraries and frameworks, which often have known vulnerabilities and expose an exploitable attack surface [22]. Vulnerabilities such as Heartbleed [38] and DROWN [20] demonstrate the criticality and impact of security issues in software dependencies. As software complexity grows with new features added on top of the old ones, tracking vulnerabilities associated with software dependencies becomes challenging for software developers and architects [35]. Information about disclosed vulnerabilities becomes fragmented across different sources [119]. Also, there are significant delays in fixing vulnerabilities in complex and large systems [75]. In our research, we witnessed unsafe software dependencies in the form of unmaintained and buggy software libraries and insecure legacy protocols.

There is a wide range of solutions to combat security issues due to dependencies, such as analyzing software compositions [108], auditing via package managers [88], automating pull requests [82], and alerting on code repositories [49]. These solutions automate the process of tracking software dependencies by collecting vulnerability information from different sources and fixing them [13, 30]. In addition, commercial products for scaling up such automation have started to emerge [78, 19]. This automation takes some of the burden of tracking and fixing vulnerabilities from the shoulders of software developers.

6.3 Empowering developer communities

Many third-party software components are developed by volunteers from the free and open-source software communities who may not have much security background. These communities may not receive any support or training to increase their security awareness. Abandoned open-source components is another concern. We encountered many orphan software repositories that are used by various software projects but abandoned by their maintainers. Despite their extensive adoption, these repositories did not have a developer community that actively contributes to and maintains them. Companies that make use of open-source software components may need to support the volunteer developer communities by providing security training and recognizing their efforts.

A recent positive trend in software industry is to embrace open-source software and to cultivate a community around it. For example, many companies have started to open-source their internal tools. They have built developer programs to provide training and certifications to open-source community members [14, 43, 79]. As discussed earlier, many companies
also offer incentives through *bug bounty* programs [42], and recognition through the *hall of fame* [52] programs. Also, they organize conferences where employees and external developer communities convene to discuss the future of the software [54, 53].

### 6.4 Systems security as a multidisciplinary problem space

Security is a multidisciplinary research area, and this thesis explores some of the disciplines. Most of the contributions in this thesis are technical in nature, such as technical security analysis of software and networks. With the technical security analysis of standards and GSMA contributions, some publications in this thesis also explore the regulatory and standards viewpoint of security. Furthermore, the thesis also covers some human and organizational aspects of security by conducting usability studies and designing a conceptual framework for communication between industry players. All these different viewpoints are needed for engineering secure systems as well as for security research.
7. Concluding Remarks

In this dissertation, we present the findings from security analysis of several different types of information systems. The diversity of topics and research methods is the main virtue of this dissertation. In fact, having diverse topics helped keep a broad scope for the research and experiment with different methodologies. The choice of topics was influenced by the author's interest and curiosity, collaboration opportunities, and trends at the time of conducting each research project. On the one hand, we analyzed specialized security technologies such as password managers, cryptocurrency wallets, VPN client software, and TPM libraries. On the other hand, we examined the security of critical information systems, such as mobile communication networks and cloud application platforms. In both cases, we tried to gain a broad understanding of the security issues and to find vulnerabilities with practical impact. From the individual vulnerabilities, we have attempted to form a big picture of the threats and the potential solutions. Our end goal has been to understand large classes of vulnerabilities and to propose solutions to them. In the threat modeling publication (i.e., Publication VI), we took an even broader view of looking at the various technical threats in the context of the entire mobile communications system.

The contributions of this dissertation include a novel adversary model to study local communication inside a computer, a conceptual framework to study mobile communications systems, the discovery of several new types of security vulnerabilities in networks and software, and usability issues faced by software developers. In the vulnerability analysis, we followed the well-established security research approach, where we first discover security weaknesses, then abstract and generalize the issues to find similar weaknesses elsewhere, and finally propose defense and mitigation solutions to fix them. The outcome of our research is that it provides new information on security issues, creates awareness about them, and helps the industry and developers to fix them. We put significant efforts into responsible disclosure of the vulnerabilities and following them up until they are fixed. Nevertheless, similar issues may still be spotted in
systems that we have not analyzed, and future efforts can continue to find and fix them.

What this dissertation attempts to accomplish is, as the title indicates, “analyzing communications and software systems security,” which required a range of approaches from the purely technical to the human factors and to the standards and regulations. Systems security is a broad theme full of deep research problems. It is also an ever evolving area where the adversaries, defenders, and academic researchers all need to continuously adapt. There are still many open and unexplored problems, and many more will arise in the future. Thus, there will be a continuous demand for further research on analyzing security as well as on building secure systems.
References


References


[23] Crofton Black. Spy companies using channel islands to track phones around the world, 2020.


[26] Matthew Braga. Inside ss7, the insecure global cell network that’s used to track phones, 2014.


[33] Joseph Cox. Criminals are tapping into the phone network backbone to empty bank accounts, 2019.

[34] Joseph Cox. We were warned about flaws in the mobile data backbone for years. now 2fa is screwed., 2019.


[89] Daniel Oberhaus. What is ss7 and is china using it to spy on trump’s cell phone?, 2018.


[96] Siddharth Prakash Rao, Silke Holtmanns, Ian Oliver, and Tuomas Aura. We know where you are! In 8th International Conference on Cyber Conflict: Cyber Power, pages 285–301. NATO CCD COE Publications, 2016.


