Introduction to the ACSAC'19 Special Issue—Part 1

The Annual Computer Security Applications Conference (ACSAC) brings together cutting-edge researchers, with a broad cross-section of security professionals drawn from academia, industry, and government, gathered to present and discuss the latest security results and topics. ACSAC's core mission is to investigate practical solutions for computer and network security technology.

The 35th Annual Computer Security Applications Conference was held in Puerto Rico on December 9–13, 2019. ACSAC 2019 especially encouraged contributions in the area of *Deployable and Impactful Security*. Deployable and impactful security solutions aim to address key real-world challenges, which may include accuracy, runtime overhead, ground-truth labeling, human aspects, usability, and energy consumption. Having the deployability and impactfulness goals motivates one to focus on solving the most critical real-world challenges, which may otherwise be ignored by the fast-moving research community. In addition, ACSAC encourages authors of accepted papers to submit software and data artifacts and make them publicly available to the entire community. Releasing software and data artifacts represents an important step toward facilitating the reproducibility of research results, and ultimately contributes to the real-world deployment of novel security solutions.

This special issue includes extended versions of papers that appeared at ACSAC 2019, focusing especially on research on computer security applications with a high potential for being deployed in real-world environments or that have already been deployed and used to implement practical defense systems.

This volume contains seven manuscripts on topics including operating systems security, software security and binary analysis, mobile device security, and web security.

In "Large-scale Debloating of Binary Shared Libraries," the authors present *Nibbler*, a debloating system that identifies and erases unused functions within dynamic shared libraries. The proposed approach works in x86-64 Linux shared libraries without requiring source code and recompilation. Nibbler reduces the size of shared libraries and the number of available functions by up to 56% and 82%, respectively, when applied in real-world binaries and the SPEC CINT2006 suite, without incurring additional run-time overhead. It also enhances current defenses like continuous code re-randomization (20% run-time improvement for Nginx) and control-flow integrity (it removes on average 75% of the available gadgets).

In "Mitigating Data-only Attacks by Protecting Memory-resident Sensitive Data," the authors address the problem of how to protect the integrity and confidentially of data in a C or C++ application automatically, even from memory disclosure vulnerabilities. The core idea of the system is to have the sensitive data (which is annotated by the developer) encrypted in memory, and the unencrypted data is only operated on in registers. In this way, according to the threat model (where an attacker cannot gain arbitrary execution, but can read memory), sensitive data is only ever in memory encrypted. The article implements this system with an LLVM pass, and presents evaluation results on several real-world applications.

ACM Reference format:
Roberto Perdisci, Martina Lindorfer, Adam Doupé, Andrea Lanzi, Alexandros Kapravelos, and Gianluca Stringhini. 2020. Introduction to the ACSAC'19 Special Issue—Part 1. *Digit. Threat.: Res. Pract.* 1, 4, Article 19e (December 2020), 3 pages.
https://doi.org/10.1145/3437251

2576-5337/2020/12-ART19e
https://doi.org/10.1145/3437251

In "Intrusion Survivability for Commodity Operating Systems," Chevalier et al. propose to switch the focus of intrusion detection from detecting attacks to the survivability of systems under attack. To achieve this, the authors propose a number of per-service responses that allow the service to continue operating under degraded capabilities that prevent attacks from succeeding (e.g., dropping privileges). The authors test this approach on a server they set up, along a number of dimensions including cost and performance. This article provides an interesting change of focus in the actions that are commonly taken when reacting to an intrusion, and the proposed ideas have the potential to influence the design of future intrusion detection systems and the actions taken by security practitioners when responding to attacks.

In "Securing Applications against Side-Channel Attacks through Resource Access Veto," Osman et al. propose *AppVeto*, a self-defense mechanism for Android apps against concurrently running apps abusing sensor data to infer sensitive information. By allowing developers to request exclusive access to sensors when performing sensitive tasks, such as password input, AppVeto prevents malicious apps from reconstructing keyboard input from motion sensors. While the authors previously demonstrated their approach using the Xposed framework for hooking and mediating access to Java APIs, they extended the approach with native code hooking to defend against native attacks as well. The authors provide their open source prototype code, which can be deployed on standard Android OS images, opening the door for further research on stricter security and privacy protections for Android apps.

In "ATFuzzer: Dynamic Analysis Framework of AT Interface for Android Smartphones," the authors propose ATFuzzer, a grammar-guided evolutionary fuzzer for dynamically testing the AT command interface in modern Android smartphones. The authors also developed an automated AT grammar extractor that allowed them to automatically extract seed AT grammars from specifications with a false-positive rate of less than 2%. The article discovers a multitude of problems on 10 smartphones from 6 different vendors: 4 problematic AT grammars through Bluetooth and 14 problematic grammars through USB interface. The discovered vulnerabilities have been reported to the corresponding vendors through responsive disclosure and resulted in two CVEs (CVE-2019-16400 and CVE-2019-16401). The ATFuzzer system, together with detailed documentation and a demo video, have been released to the public.

In "Cookies from the Past: Timing Server-Side Request Processing Code for History Sniffing," the authors present a new privacy attack where an attacker can understand what websites a victim has visited (a history sniffing technique). The core idea of this approach is the idea that the server-side code of a web application will execute different code paths in the following situations: never visited, visited, and logged in (based on the server-side code analyzing the cookies), and therefore an attacker can use the time of a response to infer these three states about the victim. The authors analyze countermeasures to this attack: and the first decision is that server-side code cannot be realistically changed so that the response time is constant in all cases. Therefore, the only counter-measure that is appropriate is a client-side defense of the SameSite cookie attribute. Then, the article studies these new defenses, outlining the evolution of the draft, browser adoption, and a large-scale study of one million websites to understand the use of this new attribute, and finally a study of websites susceptible to this attack.

Finally, in the extended version of "Will You Trust This TLS Certificate? Perceptions of People Working in IT," Ukrop et al. study how IT professionals, recruited from attendees of an industrial IT conference, deal with certificate validation errors. The authors further investigate how a different wording of error messages and documentation influences the decision making. Compared to other studies focusing on end users, this study provides interesting insights about IT experts, whose decisions in turn might also influence the security of end users. Finally, the article presents valuable lessons learned and guidelines when performing user studies on topics such as certificate validation and interpreting error messages.

As Associate Editors for this special issue, we are very pleased that the authors of the above articles have significantly extended and improved their ACSCA'19 publications, and that many of them have released their proof-of-concept software to the public to foster the reproducibility of their research results.

We thank the authors, reviewers, and ACSAC'19 program committee members who have contributed to selecting the articles that appear in this special issue. We also thank the DTRAP Co-Editors-in-Chief and the ACM for the opportunity to work on this special issue.

Roberto Perdisci
Martina Lindorfer
Adam Doupé
Andrea Lanzi
Alexandros Kapravelos
Gianluca Stringhini
Guest Editors