A CI-based Auditing Framework for Data Collection Practices

Athina Markopoulou
University of California, Irvine
athina@uci.edu

Rahmadi Trimananda
University of California, Irvine
rtrimana@uci.edu

Hao Cui
University of California, Irvine
cuih7@uci.edu

ABSTRACT

Apps and devices (mobile devices, web browsers, IoT, VR, voice assistants, *etc.*) routinely collect user data, and send them to first- and third-party servers through the network. Recently, there is a lot of interest in (1) auditing the actual data collection practices of those systems; and also in (2) checking the consistency of those practices against the statements made in the corresponding privacy policies. In this paper[1], we argue that the contextual integrity (CI) tuple can be the basic building block for defining and implementing such an auditing framework. We elaborate on the special case where the tuple is partially extracted from the network traffic generated by the end-device of interest, and partially from the corresponding privacy policies using natural language processing (NLP) techniques. Along the way, we discuss related bodies of work and representative examples that fit into that framework. More generally, we believe that CI can be the building block not only for auditing at the edge, but also for specifying privacy policies and system APIs. We also discuss limitations and directions for future work.

ACM Reference Format:
Athina Markopoulou, Rahmadi Trimananda, and Hao Cui. . A CI-based Auditing Framework for Data Collection Practices. In . ACM, New York, NY, USA, 5 pages.

1 THE PROBLEM SPACE

Personal data are routinely collected on end devices (browsers, mobile and IoT devices, smart TVs, VR devices, *etc.*) and shared with many first- and third-party entities, without providing much transparency or control to users. Recently, increased public awareness has led to data protection legislation, such as the GDPR, CCPA/CPRA, and other state or sector-specific data protection laws. These laws state rights of consumer or citizens, and duties of entities that collect, share, and use personal data. Government agencies, such as the U.S. Federal Trade Commission (FTC), take initiatives to enforce those regulations. Their efforts are amplified by non-profits, privacy-advocates, and academics who report the results of their investigation on data collection practices and violations. These developments have pushed tech companies, which were previously lacking incentives to self-regulate, towards the right direction, *i.e.,* to become increasingly more transparent about their data collection practices and apply other good practices as well.

The Gap between Systems and Laws. However, there is still a significant gap between the practices of tech companies, and the formulation and enforcement of privacy laws. First, from a software developer's perspective, it is often difficult to ensure compliance with all laws and regulations due to the complexity of their own system as well as their dependence on third party libraries, platforms, and other parts of the ecosystem that they have no control over. Second, policymakers need technical input to write relevant and enforceable policies. Sometimes the requirements in the laws do not directly map to a system specification that can be implemented and audited. Furthermore, the technology itself evolves rapidly and often renders privacy laws obsolete. The general problem of co-designing privacy-preserving systems along with consistent privacy laws—let alone auditing tools—is already quite daunting. Progress is made on narrow notions such as the meaning of "singling out" a user [6], and this is an area of increasing research interest.

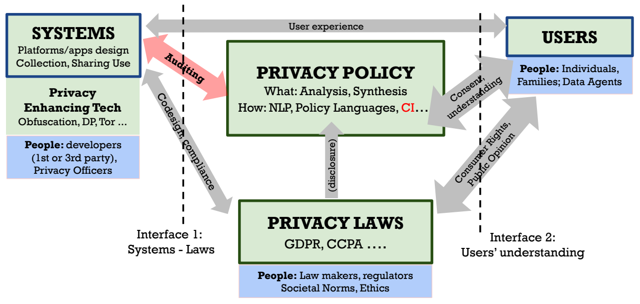

SYSTEMS
Platforms/apps design
Collection, Sharing Use
Privacy Enhancing Tech
Obfuscation, DP, Tor ...
People: developers (1st or 3rd party), Privacy Officers
User experience
USERS
People: Individuals, Families; Data Agents
Auditing
PRIVACY POLICY
What: Analysis, Synthesis
How: NLP, Policy Languages, CI...
Consent, understanding
Codesign, compliance
(disclosure)
Consumer Rights, Public Opinion
PRIVACY LAWS
GDPR, CCPA ....
People: Law makers, regulators
Societal Norms, Ethics
Interface 1: Systems - Laws
Interface 2: Users' understanding

Figure 1: The privacy problem space involves computer systems, privacy laws, and users. An important component interfacing with all three is the privacy policy of the system—we presented an earlier version of this diagram in [4]. In this paper, we focus on the part highlighted in red: auditing the actual data collection practices of the system and the statements made in the corresponding privacy policy.

Transparency and Privacy Policies. Although the general problem of technology-law interface is quite complex and difficult to tackle, there have been efforts and progress on developing methodologies to solve a more narrow problem: *transparency.* At the very least, most privacy laws require that entities disclose their data collection, sharing, and use practices. For example, entities must disclose *what they collect, with whom they share it, for what purposes, etc.* Furthermore, most privacy laws also require that the user is notified and consents to those practices. Software systems and services typically fulfill their "notice-and-consent" obligation through a *privacy policy* document that they provide to inform their users about their practices. Then, the user can choose to opt in or opt out before using the product or service. Privacy policy documents are legally binding and they are the focus of this paper.[2]

An entity's written privacy policy is the interface between privacy laws and system design/implementation, as well as communication with the user; this is depicted in Figure 1. However, while having *a* privacy policy is necessary, it is not sufficient to provide transparency due to the following challenges:

1) A privacy policy must be written so that it meets the minimum requirements in order to be compliant with the privacy laws.

[1]This work is supported by NSF Award 1956393.

[2]There may be additional disclosures and privacy notices via blogs, press releases, websites, *etc.*, which further communicate to the public how an organization collects and processes personal data, and what rights it provides.

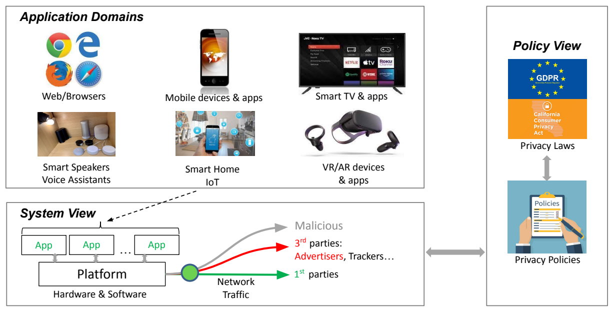


Figure 2: Data collection at the edge by apps, devices, and platforms. The outgoing network traffic provides a unique vantage point for monitoring the actual information flow.

2) A privacy policy must accurately and comprehensively describe the data collection practices implemented by the system and any third-party libraries the system uses.
3) A privacy policy should be (i) comprehensive in its coverage of the system functionality and law requirements, (ii) self-consistent, *i.e.,* without contradicting statements, and present information to the user in a way that is easily understood.

Auditing Data Collection at the Edge. To further narrow down the scope of the problem, let us consider Figure 2. It depicts various end-devices and platforms that users interact with, including mobile devices, browsers, smart TVs, VR devices, IoT devices, *etc.*. These are different platforms that enable their respective apps. In all these app ecosystems, personal data are collected on devices (from the apps, platforms, and third-party libraries), and are sent over the Internet to first- and third-party entities for functional, advertising and tracking services (ATS), and many other purposes. The research community has followed different types of broad approaches for auditing data collection practices at the edge:

1) A large body of work obtains and analyzes the *actual* information flow observed into and out of an app, device, or platform using a range of techniques, *i.e.,* static [8] and dynamic analysis [7], and network traffic analysis [14, 16–20, 27].
2) Another large body of work extracts the *intended/declared* information flow. This is mostly done by analyzing the privacy policy of the system; it originally relied on experts reading the policies, but is getting increasingly automated using NLP [1, 9]. Another source for extracting the *intended/declared* information flow is through permissions [12].
3) There is also a third body of work that checks the *consistency* between the intended/declared information flow in the privacy policy, and the actual behavior found in the system (*e.g.,* in the network traffic [2]).

Our key observation is that in all the above cases, the extracted information is *part* of the parameters of the CI-tuple:[3]

$$\boxed{\langle sender, recipient, data\ type, [subject], transmission\ principle\rangle}$$

By observing the actual behavior of the system or its declared behavior in its privacy policy, one can observe and automatically extract meaningful values from each CI parameter as summarized in Figure 3. The three aforementioned bodies of work typically

[3]When analyzing information flow at the edge, the "subject" in the CI-tuple is the user of the device/app, thus will be omitted for lack of space in this paper.

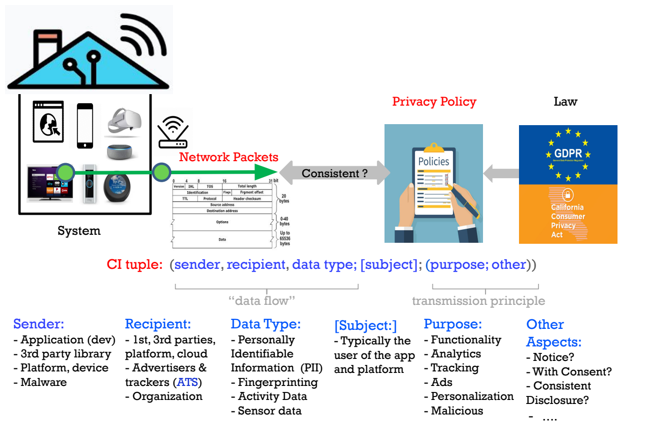


Figure 3: Extracting (parts of) the CI tuple from the network traffic and the privacy policy.

obtain and report part of the tuple, often without recognizing that it fits into the CI framework [15].

Position. We believe that the CI-tuple is well-suited to be the central building block (data structure) for specifying and auditing data collection practices at the edge, together with their privacy policies. Not only that it is intuitive and interpretable, but it also lends itself to automation and large-scale auditing. Representative examples are discussed in Section 2. Limitations and possible extensions are discussed in Section 3.

2 CI-BASED AUDITING

2.1 Network Traffic Monitoring

Apps, platforms, and third-party libraries running on a device can collect personal data and send them to remotes server over the Internet, as depicted in Figures 2 and 3. Monitoring the network traffic coming out of the device provides a unique vantage point to observe data collection: *who* collects *what*, and *where* it is sent.

Intercepting and inspecting network traffic, can be done on the device itself (examples on Android include our own AntMonitor [18] and Lumen [16]), or using a proxy [13]. Packet traces are collected, and typically stored and analyzed offline. By observing parts of the packets that are sent in the clear, it is trivial to extract the *recipient* of the information flow, *e.g.,* through destination domains (DNS queries), IP addresses, reverse DNS lookups, *etc.* It is also trivial to extract further information about the organization/entity a destination domain belongs to (*e.g.,* using "whois" or other lookup methods), and classify the domain as first- (*e.g.,* if the domain matches or partially contains the name of the entity), or third-party (*e.g.,* it is categorized as an ATS domain based on well-known blocklists). This labeling as first- vs. third-party also indicates the *purpose* of data collection. Furthermore, when network traffic collection is done on the device [16, 18], it is also trivial to extract the *sender* (*i.e.,* app, platform, or third-party library). It is more difficult to extract the *data types* sent in network packets, since the packets are typically encrypted. While this encryption can be bypassed, it is oftentimes not clear what to look for in the payload of a decrypted packet; the payload usually contains key-value pairs, whose meaning often cannot be interpreted precisely. In summary, from collected network packet traces at the edge, we can extract several parameters of the CI-tuple:

$$\boxed{F' = \langle sender = platform/app, destination, data\ type, purpose\rangle}$$

Several researchers in the Internet measurement community have performed network traffic analysis on different platforms to characterize their data collection and sharing practices focusing on ATS. For example, just within our group, we have systematically tested the apps and inspected the traffic generated by mobile devices and their apps [19, 20], smart TV platforms and their apps [26, 27], IoT devices [24], VR headsets [23], and most recently smart speakers [10]. We have also characterized and reported the aforementioned CI parameters. Many other groups have done similar studies for these platforms; examples include [3, 11, 14, 16, 17], but the list is by no means exhaustive—this is an active field.

2.2 Privacy Policy Analysis

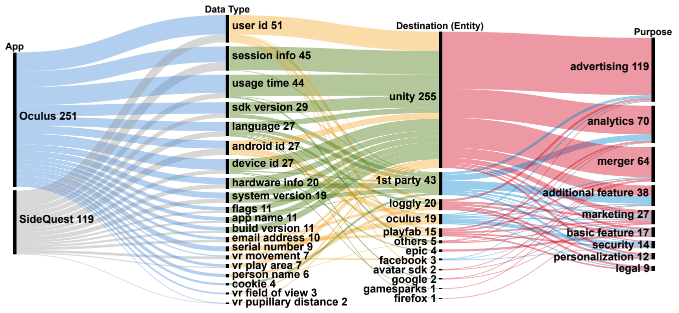
Privacy policy and consistency analysis is becoming increasingly automated using NLP, and applied across various app ecosystems [1, 2, 9, 22, 28–30]; this is also a large body of work to properly review here. Instead, we describe state-of-the-art tools, namely PolicyLint [1], PoliCheck [2], and Polisis [9] that demonstrate what is possible w.r.t. CI. They all apply NLP on: (1) the text of privacy policies to extract and analyze data collection and sharing statements; and (2) compare them against the actual data flows found in network traffic (*flow-to-policy* consistency analysis).

PolicyLint [1] provides an NLP pipeline that takes a sentence as input. For example, it takes *"We may collect your email address and share it for advertising purposes"*, and extracts the collection statement "(entity: *we*, action: *collect*, data type: *email address*)". More generally, PolicyLint takes the app's privacy policy text, parses sentences, performs NLP techniques, and eventually extracts data collection statements defined as the tuple $P = \langle app, data\ type, entity\rangle$; *app* is the sender and *entity* is the recipient organization/entity performing an *action* ("collect" or "not collect") on the *data type*, and outputs:

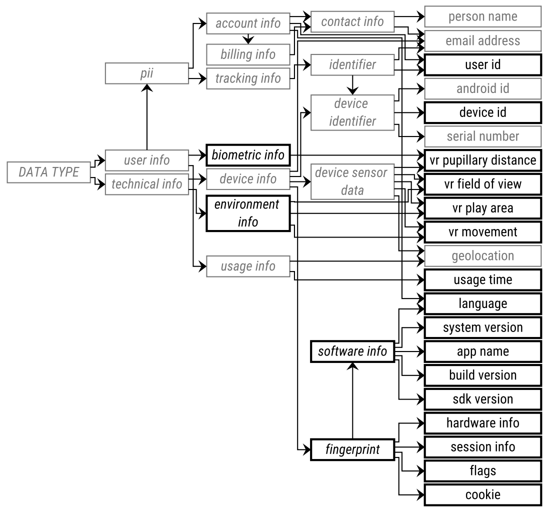
$$\boxed{P = \langle sender = platform/app, recipient = entity, data\ type\rangle}$$

PoliCheck [2] takes the app's "data flows" (extracted from the network traffic and defined as $F = \langle data\ type, entity\rangle$ and compares it against the stated P for consistency. PoliCheck classifies the disclosure of F as *clear* (if the data flow exactly matches a collection statement), *vague* (if the data flow matches a collection statement in broader terms), *omitted* (if there is no collection statement corresponding to the data flow), *ambiguous* (if there are contradicting collection statements about a data flow), or *incorrect* (if there is a data flow for which the collection statement states otherwise).[4]

Consistency analysis between the "data flows" (F or its augmented version F') found in network traffic and collection statements (P or its augmented version P') rely on pre-built ontologies and synonym lists. These are used to match (i) the data type and destination that appear in each F with (ii) any instance of P that discloses the same (or a broader) data type and destination[5].



(a) P' tuples: augmenting P (from PoliCheck) with *purpose* (from Polisis).



(b) VR-specific data ontology

Figure 4: Case study: auditing network traffic and privacy policies of the most popular Oculus VR apps; see [23].

PolicyLint and PoliCheck were originally developed for mobile apps and later applied to Alexa skills [12] and Oculus VR apps [23]. In [23], we developed Oculus VR-specific ontologies; the data ontology is shown on Figure 4(b). Furthermore, we improved several aspects of PoliCheck and augmented the P tuple with *purpose*, which we were able to extract from the privacy policy using Polisis [9]. The augmented tuples are visually depicted on Figure 4(a):

$$\boxed{P' = \langle sender = platform/app, recipient = entity, data\ type, purpose\rangle}$$

Purpose/use is an important dimension of the transmission principle (TP) parameter in CI and there are automated ways to automatically extract purpose from privacy policies. Other state-of-the-art methods beyond Polisis [9] include the following. MobiPurpose [11] inferred data collection purposes of mobile apps using network traffic and app features (*e.g.,* URL paths, app metadata, domain name, *etc.*). PurPliance [5] automates the inference of data collection purposes introduced in MobiPurpose, extracts purposes from the privacy policy, and performs flow-to-policy consistency analysis that includes checking the consistency of these purposes.

[4]Following PoliCheck's terminology [2], these five types of disclosures can be grouped into two groups: *consistent* (clear and vague disclosures) and *inconsistent* (omitted, ambiguous, and incorrect) disclosures. For consistent disclosures, there is a statement in the policy that matches the data type and entity, either clearly or vaguely.

[5]For example see Figure 4(b): "email address" is a special case of "contact info" and, in turn, of "pii". There is a clear disclosure *w.r.t.* data type if the "email address" is found in a data flow and a collection statement. A vague disclosure is declared if the "email address" is found in a data flow and a collection statement that uses the term "pii" is in the privacy policy. An omitted disclosure means that "email address" is found in a data flow, but there is no mention of it (or any of its broader terms) in the privacy policy.

2.3 Extracting the CI tuple

A key observation is that existing methods for monitoring and reporting information flow from an end device already extract (a subset or all of) the CI parameters in the CI-tuple. However, they currently do it in an ad-hoc way, often without realizing that these methods fit into the CI framework, with a few exceptions [21]. Furthermore, this extraction can be done through an *automated* pipeline: automatically testing the behavior of a large number of apps, collecting network traffic, extracting the "data flows" P found in the traffic, performing NLP techniques to analyze privacy policies, and extracting the collection and sharing statements F. Extracting the well-defined parameters of the CI-tuple, *i.e.,* $P' = \langle sender = platform/app, recipient = entity, data\ type, subject = user\rangle$, is well understood by now. Even some values of the more elusive TP can be extracted automatically. For example in [23], we were able to extract the following dimensions of TP: (1) *consistency, i.e.,* whether actual data flows extracted from network traffic agree with the corresponding statements made in the privacy policy; (2) *purpose*, extracted from privacy policies and confirmed by destination domains (*e.g.,* whether they are ATS); and (3) the presence of "notice-and-consent", while testing the apps.

We believe that the CI-tuple can provide a unifying data structure, and interface for auditing both the intended/declared and actual end-device behaviors, as well as the consistency between the two.

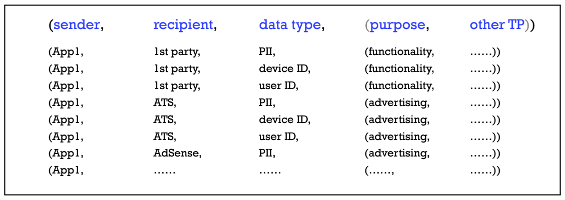
3 OPEN PROBLEMS & FUTURE DIRECTIONS

We have argued that parts for the CI-tuple is already used in practice for auditing systems and their policies, and that the CI-tuple should be proposed in a more intentional and unifying way as the data structure for auditing data collection and sharing practices. However, there are still open conceptual problems (beyond just defining TP) to address, before CI can realize this potential.

Q1: Dealing with hierarchical parameters. The CI-tuple is by definition "flat", as depicted on Figure 5(a). However, when we audit a device (both from the system and privacy policy perspectives), we collect multiple such tuples capturing different information flows (one per packet or per collection statement, respectively). The values of the same CI-parameter can vary across different tuples. How should we summarize and process this information?

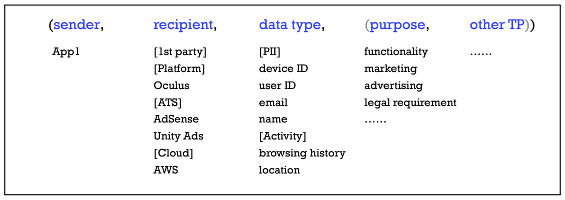
One option is to consider the values for each CI parameter as unrelated, essentially a laundry list of possible values, as in Figure 5(b). Shvartzshnaider *et al.* analyzed Facebook's privacy policy [21] and coined the term "bloated" policy. Considering the actual CI-tuple in its full context, as in Figure 5(a), is a more precise disclosure. However, many privacy policies today (even the good ones like Unity's [25]) are organized in sections containing bulleted lists (*e.g.,* what is collected, with whom it is shared, for what purpose, *etc.*) to match what the law dictates and to also be more readable.

There is one more complication: possible values of each CI parameter can be organized in an ontology: a conceptual example is shown on Figure 5(c) and an example based on real data is shown on Figure 4(b). This hierarchy is not only intuitive, but also necessary for checking the consistency between a value observed in the network traffic and what is declared in the policy. Without an ontology, collecting an "email" would be inconsistent with the privacy policy that states the collection of "PII": the former is clearly subsumed by the latter. We believe that developing a concept of a "hierarchical" CI-tuple would be necessary for auditing.



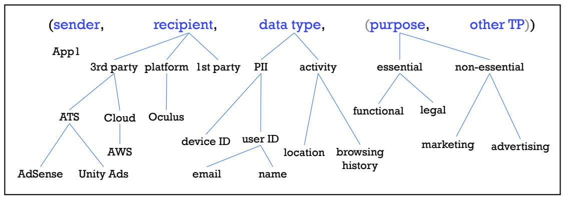
(sender,	recipient,	data type,	(purpose,	other TP))
(App1,	1st party,	PII,	(functionality,	......))
(App1,	1st party,	device ID,	(functionality,	......))
(App1,	1st party,	user ID,	(functionality,	......))
(App1,	ATS,	PII,	(advertising,	......))
(App1,	ATS,	device ID,	(advertising,	......))
(App1,	ATS,	user ID,	(advertising,	......))
(App1,	AdSense,	PII,	(advertising,	......))
(App1,	......	......	(......,	......))

(a) "Flat" CI tuples



(sender,	recipient,	data type,	(purpose,	other TP))
App1	[1st party]	[PII]	functionality	......
	[Platform]	device ID	marketing	
	Oculus	user ID	advertising	
	[ATS]	email	legal requirement	
	AdSense	name	......	
	Unity Ads	[Activity]		
	[Cloud]	browsing history		
	AWS	location		

(b) "Bloated" CI tuples



(c) "Hierarchical" CI-tuples (*i.e.,* with ontologies).

Figure 5: Open question: how to extract and summarize (a large number of) CI tuples? Note: the "subject" parameter is implicitly the user of the device or app, and it is omitted.

Q2: How to combine parameters extracted from different sources into a single CI-tuple. Different sources (*e.g.,* network traffic, dynamic analysis, NLP on privacy policy, permissions, *etc.*) can reveal some or all the parameters of the CI-tuple. How does one merge that information from different sources into a single CI-tuple? If the value of the same CI parameter obtained from different sources is consistent, it can help link the corresponding tuples and add dimensions that were possibly missing.[6] But how should one deal with inconsistencies? In [23], we included consistency as part of the TP parameter, but there may be other ways worth exploring.

Q3: Reactive vs. proactive use of CI. What we proposed so far is a *reactive/diagnostic* use of CI for auditing, namely to extract and report the tuple, so as to characterize the appropriateness of the information flow. In addition, we believe that CI can play a *proactive* role in defining the specification of privacy policies and system APIs. The sections currently found in privacy policies (see Unity's privacy policy [25] as an example) can follow the structure of the CI-tuple. Furthermore, platforms and their apps can be designed to provide APIs to return the CI-tuple for auditing purposes.

Finally, revisiting the bigger picture in Figure 1, using the CI-tuple to co-design privacy policies and systems can also be useful in other interactions, such as in the communication with users and in interfacing with privacy laws (possibly through the TP).

[6] This is what we did in the example in Figure 4(a): we took the flow-to-policy analysis result from PoliCheck that maps data flows $\langle destination, data\ type\rangle$ from network traffic to policy statements, and mapped each statement to its corresponding policy segment. We then used Polisis to extract purposes from each segment, and reported the augmented tuple.

REFERENCES

[1] Benjamin Andow, Samin Yaseer Mahmud, Wenyu Wang, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Tao Xie. Policylint: Investigating internal privacy policy contradictions on google play. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 585–602. USENIX Association, August 2019.

[2] Benjamin Andow, Samin Yaseer Mahmud, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Serge Egelman. Actions speak louder than words: Entity-sensitive privacy policy and data flow analysis with policheck. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 985–1002. USENIX Association, August 2020.

[3] Noah Apthorpe, Danny Yuxing Huang, Dillon Reisman, Arvind Narayanan, and Nick Feamster. Keeping the smart home private with smart (er) iot traffic shaping. *arXiv preprint arXiv:1812.00955*, 2018.

[4] Athina Markopoulou and Shomir Wilson. Breakout session on "Privacy, Policy and People", NSF SaTC PI Meetting 2022. https://cps-vo.org/group/satc-pimtg22/breakouts.

[5] Duc Bui, Yuan Yao, Kang G Shin, Jong-Min Choi, and Junbum Shin. Consistency analysis of data-usage purposes in mobile apps. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2824–2843, 2021.

[6] Aloni Cohen and Kobbi Nissim. Towards formalizing the gdpr's notion of singling out. *Proceedings of the National Academy of Sciences*, 117(15):8344–8352, 2020.

[7] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*. USENIX Association, October 2010.

[8] William Enck, Damien Octeau, Patrick McDaniel, and Swarat Chaudhuri. A study of android application security. In *20th USENIX Security Symposium (USENIX Security 11)*, pages 315–330. USENIX Association, August 2011.

[9] Hamza Harkous, Kassem Fawaz, Rémi Lebret, Florian Schaub, Kang G. Shin, and Karl Aberer. Polisis: Automated analysis and presentation of privacy policies using deep learning. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 531–548. USENIX Association, August 2018.

[10] Umar Iqbal, Pouneh Nikkhah Bahrami, Rahmadi Trimananda, Hao Cui, Alexander Gamero-Garrido, Daniel Dubois, David Choffnes, Athina Markopoulou, Franziska Roesner, and Zubair Shafiq. Your echos are heard: Tracking, profiling, and ad targeting in the amazon smart speaker ecosystem. *arXiv preprint arXiv:2204.10920*, 2022.

[11] Haojian Jin, Minyi Liu, Kevan Dodhia, Yuanchun Li, Gaurav Srivastava, Matthew Fredrikson, Yuvraj Agarwal, and Jason I Hong. Why are they collecting my data? inferring the purposes of network traffic in mobile apps. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, volume 2, pages 1–27. ACM, 2018.

[12] Christopher Lentzsch, Sheel Jayesh Shah, Benjamin Andow, Martin Degeling, Anupam Das, and William Enck. Hey alexa, is this skill safe?: Taking a closer look at the alexa skill ecosystem. In *28th Annual Network and Distributed System Security Symposium (NDSS 2021)*. The Internet Society, 2021.

[13] mitmproxy. Mitmproxy Project. https://mitmproxy.org/.

[14] Hooman Mohajeri Moghaddam, Gunes Acar, Ben Burgess, Arunesh Mathur, Danny Yuxing Huang, Nick Feamster, Edward W Felten, Prateek Mittal, and Arvind Narayanan. Watching you watch: The tracking ecosystem of over-the-top tv streaming devices. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 131–147. ACM, 2019.

[15] Helen Nissenbaum. *Privacy in Context - Technology, Policy, and the Integrity of Social Life.* Stanford University Press, 2010.

[16] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, Christian Kreibich, and Phillipa Gill. Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. In *25th Annual Network and Distributed System Security Symposium (NDSS 2018)*. The Internet Society, 2018.

[17] Jingjing Ren, Ashwin Rao, Martina Lindorfer, Arnaud Legout, and David Choffnes. Recon: Revealing and controlling pii leaks in mobile network traffic. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys 16)*, page 361–374. ACM, 2016.

[18] Anastasia Shuba, Anh Le, Emmanouil Alimpertis, Minas Gjoka, and Athina Markopoulou. Antmonitor: A system for on-device mobile network monitoring and its applications. *arXiv preprint arXiv:1611.04268*, 2016.

[19] Anastasia Shuba and Athina Markopoulou. Nomoats: Towards automatic detection of mobile tracking. In *Proceedings on Privacy Enhancing Technologies*, volume 2020, pages 45–66. Sciendo, 2020.

[20] Anastasia Shuba, Athina Markopoulou, and Zubair Shafiq. Nomoads: Effective and efficient cross-app mobile ad-blocking. In *Proceedings on Privacy Enhancing Technologies*, volume 2018, pages 125–140. Sciendo, 2018.

[21] Yan Shvartzshnaider, Noah Apthorpe, Nick Feamster, and Helen Nissenbaum. Going against the (appropriate) flow: a contextual integrity approach to privacy policy analysis. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 7, pages 162–170, 2019.

[22] Rocky Slavin, Xiaoyin Wang, Mitra Bokaei Hosseini, James Hester, Ram Krishnan, Jaspreet Bhatia, Travis D. Breaux, and Jianwei Niu. Toward a framework for detecting privacy policy violations in android application code. In *Proceedings of the 38th International Conference on Software Engineering (ICSE 16)*, pages 25–36. ACM, 2016.

[23] Rahmadi Trimananda, Hieu Le, Hao Cui, Janice Tran Ho, Anastasia Shuba, and Athina Markopoulou. OVRseen: Auditing Network Traffic and Privacy Policies in Oculus VR. In *Proceedings of the 31st USENIX Security Symposium (USENIX Security 22)*, 2022.

[24] Rahmadi Trimananda, Janus Varmarken, Athina Markopoulou, and Brian Demsky. Packet-level signatures for smart home devices. In *Network and Distributed Systems Security (NDSS) Symposium*, volume 2020, 2020.

[25] Unity Technologies. Unity privacy policy. https://unity3d.com/legal/privacy-policy, 2022.

[26] Janus Varmarken, Jad Al Aaraj, Rahmadi Trimananda, and Athina Markopoulou. FingerprinTV: Fingerprinting Smart TV Apps. *Proceedings on Privacy Enhancing Technologies*, 2022(3), 2022.

[27] Janus Varmarken, Hieu Le, Anastasia Shuba, Athina Markopoulou, and Zubair Shafiq. The tv is smart and full of trackers: Measuring smart tv advertising and tracking. In *Proceedings on Privacy Enhancing Technologies*, volume 2020, pages 129–154. Sciendo, 2020.

[28] Xiaoyin Wang, Xue Qin, Mitra Bokaei Hosseini, Rocky Slavin, Travis D. Breaux, and Jianwei Niu. Guileak: Tracing privacy policy claims on user input data for android applications. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 37–47. IEEE, 2018.

[29] Sebastian Zimmeck and Steven M. Bellovin. Privee: An architecture for automatically analyzing web privacy policies. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 1–16. USENIX Association, August 2014.

[30] Sebastian Zimmeck, Ziqi Wang, Lieyong Zou, Roger Iyengar, Bin Liu, Florian Schaub, Shomir Wilson, Norman M. Sadeh, Steven M. Bellovin, and Joel R. Reidenberg. Automated analysis of privacy requirements for mobile apps. In *24th Annual Network and Distributed System Security Symposium (NDSS 2017)*. The Internet Society, 2017.