The Trusted CI Webinar Series is supported by National Science Foundation grant #1547272.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the NSF.
Ancile: Enhancing Privacy for Ubiquitous Computing with Use-based Privacy

Jason Waterman
Vassar College

NSF Award: 1642120

Collaborators:
Cornell Tech: Eugene Bagdasaryan, Deborah Estrin
Cornell University: Nate Foster, Fred Schneider
Pomona College: Eleanor Birrell
Vassar College: Griffin Berlstein
We generate significant amounts of personal data on a daily basis

- Web browsing
- Pages visited
- Ad engagement
- Smartphones
- Messaging
- Location
- Call History
- IoT Devices
- Biometrics
- Sensor data
And that data is valuable... perhaps *too* valuable...

Despite privacy outrage, AccuWeather still shares precise location data with ad firms.

New tests reveal that while one privacy-invading feature was removed in an app update, the app still shares precise geolocation coordinates with advertisers.

Your Apps Know Where You Were Last Night, and They’re Not Keeping It Secret

Dozens of companies use smartphone locations to help advertisers find you. They say it’s anonymous, but the data shows how it is.

FaceApp under fire as lawmakers ask FBI to review viral app

Facebook allows advertisers to target you based on your shadow profile.
However... you might want your data included

Personal Health and Lifestyle
• Stanford Medicine Apple Heart Study
• Biometric Tracking and Analysis
• Personalized Nutrition

Recommender and Prediction Systems
• Entertainment recommendation
• Intelligent autocomplete
• Digital assistants

Netflix knows that I like John Mulaney
From Ubiquitous Data to Data Driven Planning
Privacy Preserving Data-Rich Applications

Want to facilitate and enable the development of privacy preserving data-rich applications

This requires us to answer three questions:

• What is privacy?
• How to we represent it?
• How do we enforce it?
What is Privacy?

• There are many ways to define privacy, we’ll work with the following definition:

Privacy is the restriction of information flows to only appropriate contexts and forms

• The preservation of privacy is then the enforcement of these restrictions

• For this to be useful, we need a model to express what is appropriate
Representing Privacy: Use-based Privacy Model

- A **reactive** approach where acceptable actions depend on a **privacy policy** attached to the data
- The context surrounding the data and the intended use matter

- **Reactivity**: the state of a privacy policy depends on
  - The history of actions on the data
  - The data itself

- The **state** of a privacy policy determines what actions are acceptable and what entities may execute them
Use-based Privacy Example

• Policy on location data: “Share my location only when I’m on campus, but fuzz my location to 100 meter circle first before sharing”
  • Use depends on context (on campus or not)
  • Restrictions change as data is transformed (allow after location is fuzzed)
  • Current policy depends on the history of events that has occurred

• Can be represented as a state transition diagram

Fetch_Location  →  On_Campus  →  Fuzz_Location(100)  →  Return
Policy Language

- Policies are specified as regular expressions over an alphabet of commands
  - Operate on data

- Specifies how a data value may be used
  - How derived values may be used

Fetch_Location.On_Campus._test_true.Fuzz_Location(100).Return
Policy Evaluation and Derivatives

• For a command \( c \), a policy \( P \) authorizes \( c \) if there exists a string \( S \) with prefix \( c \) such that \( S \in L(P) \)
  • \( L(P) \) denotes the set of strings generated by the regular expression policy \( P \)
• If the command is authorized, we advance the policy by taking the Brzozowski derivative
  • \( D(P_x, c) \) where \( P_x \) is the policy for data \( x \) and command \( c \)
  • \( P_x = Fetch\_Location.\_On\_Campus.\_test\_true.\_Fuzz\_Location(100).\_Return \)
  • \( c = Fetch\_Location \)
  • \( D(P_x, c) = On\_Campus.\_test\_true.\_Fuzz\_Location(100).\_Return \)
Outline

• What is privacy?

• How to we represent it?

• How do we enforce it?
Ancile: A system for Policy Enforcement

• Run-time monitor positioned between an application and a user’s sensitive data to enforce a privacy policy
Ancile Data Providers

• External third-party entities that collect and store data on behalf of the user
• Users are able to delegate data access authority to Ancile (e.g., via OAuth2)
• We’ve focused on location data for our example applications
Ancile Users

• Users register with Ancile
  • Can add data providers and policies via web dashboard
• Applications provide Ancile users with their requests
Ancile Applications

- Consume data to provide some service
- Make https requests to Ancile to access data
- Provides a program which runs in Ancile’s trusted environment
- Applications are not trusted to be policy compliant
Ancile Overview (1)

- Applications submit requests to Ancile
  - Contains a program to be executed in Ancile’s trusted environment
  - Credentials to authenticate the application to Ancile
- Program runs in a restricted environment
Ancile Overview (2)

• Ancile fetches data from data provider
  • Upon ingress into Ancile, it is tagged with its policy
  • Data and policy travel together as a DataPolicyPair

![Ancile Diagram]

- AncileWeb
- AncileCore
- AncileLib
  - Data processing
  - enforced by policies
- User
  - access to data sources + policies
- Application
  - derived data
  - program
  - user credentials
  - sensitive data
- Data Providers
  - Data processing
  - enforced by policies
  - commands
Ancile Overview (3)

• Ancile executes the submitted application program
  • Each command in the program is checked for policy compliance
Ancile Overview (4)

- Output data is sent to the application
  - Only if all commands in the program were authorized by the policy
Ancile Overview (5)

- When the program finishes or is terminated for policy violation
  - All data is deleted from Ancile
Ancileweb

- Web dashboard for interacting with Ancile

- Manages Ancile user accounts
  - View, delete, and register data providers
  - View, edit, and delete policies

- Applications and Admins have their own dashboard
AncileWeb Policy Visualizer

• Allow users and policy administrators to visualize and edit polices

• Can edit and visualize policies in real-time
AncileLib

- Library of privileged commands provided by Ancile
  - Used by applications to implement programs
  - Commands run inside of Ancile’s trusted environment
  - Policy checked by AncileCore before execution
  - Have access to data

- Commands must be vetted and approved before being made available to applications

- Four types of commands:
  - External – Fetch data from external sources
  - Condition – Evaluate a predicate on a data value
  - Transformation – Compute a derived value for data
  - Return – Send a value back to the application
AncileLib External Commands

• Receive access tokens from AncileWeb
• Requests data from a data provider
• Checked for policy compliance before command is executed
• Data returned from this command is tagged with a policy
AncileLib Condition Commands

• Evaluate a predicate on a data value
• Allows for branching conditions in policies
• Makes use of two internal Ancile commands
  • _test_false
  • _test_true
AncileLib Transformation Commands

• Transforms data and updates policy
• Commands can take arguments
  • `fuzz_location(radius = 100)`
• If specified in a policy, arguments are checked for compliance
• All comparisons operations are supported
AncileLib Return Commands

• Sends a data to an application
• Once a data value is returned, we make no further guarantees how it will be used by the application
AncileCore

• Ancile’s reference monitor
  • Receives and executes programs on behalf of applications
  • Enforces use restrictions

• Three components of an application
  Ancile data request
  1. Application token: the secret used to authenticate the application to Ancile
  2. Users: the users the application is requesting data for
  3. Program: the computation to be executed within Ancile
Ancile Programs

• Submitted Ancile programs are written in Python
• After AncileCore verifies the application request
  • Compiles the submitted program using RestrictedPython
    • Prevents arbitrary functions and libraries from being called
    • Restricts program to only run Ancile commands
    • Prevents access to restricted object fields
• Ancile operates on policy tagged values known as DataPolicyPairs
  • Contain two restricted fields
    • _data
    • _policy
• Ancile commands are written as privileged Python functions and are able to see and operate on these restricted fields
Ancile Program Example

Program:

dpp1 = get_last_location(user="jawaterman")

if in_geofence_cond(data=dpp1, geofence="Vassar"):  
    dpp2 = fuzz_location(radius=100)  
    return_to_app(data=dpp2)

Policy:

get_last_location
.in_geofence_cond(geofence="Vassar")
._test_true.fuzz_location(radius=100).return_to_app
  + _test_false
Context-Dependent Policies

• Ancile can also express authorizations that depend on the external state (i.e., another data source)

• Example: Roaming Office Hours
  • Application designed for TAs or Professors who wish to hold regular office hours at irregular locations
  • Context-dependent: location data is only released if the desired conditions are true
Roaming Office Hours Program

Program:

loc_dpp = get_last_location(user="jawaterman")
cal_dpp = get_calendar_events(user="jawaterman")

if in_geofence_cond(data=loc_dpp, geofence="Vassar"):
    if event_occuring_cond(data=cal_dpp,
        event_name="Office Hours",
        dependent=loc_dpp):
        return_to_app(data=loc_dpp)
Roaming Office Hours Policy

Ancile Policy

get_current_events

event_occuring_cond
  event_name = Office Hours
  user = jawaterman@vassar.edu
  data_source = Google_Cal

get_last_location

in_geofence_cond
  geofence = Vassar

OR

_test_true

event_occuring_cond
  event_name = Office Hours
  user = jawaterman@vassar.edu
  data_source = Google_Cal

OR

_test_false

return_to_app
Aggregate Transformations

• To support commands that take multiple DataPolicyPairs as arguments, we support aggregate transformations.

• Aggregations take multiple DataPolicyPairs and return a single DataPolicyPair:
  • Policy is synthesized as the intersection of the derivative of the individual inputs.

• Example: `evaluate_quorum` in the following pipeline.
Group Study Application

• A slackbot that helps small groups of users collaborate by enabling impromptu face-to-face meetings
• It maintains a list of group members and periodically checks whether a quorum of the group is on site
  • If so it notifies all group members via a slack message
Group Study Example

Program:

dpp_A = fetch_location(user="userA")
dpp_B = fetch_location(user="userB")
dpp_C = fetch_location(user="userC")

gf_A = compute_geofence(data=dpp_A, geofence="Library")
gf_B = compute_geofence(data=dpp_B, geofence="Library")
gf_C = compute_geofence(data=dpp_C, geofence="Library")

dpp_aggr = evaluate_quorum(data=[gf_A, gf_B, gf_C], threshold_percent=100)

return_to_app(data=dpp_aggr)

Policy:

fetch_location
  .compute_geofence(geofence="Library")
  .evaluate_quorum(threshold_percent=100, users=["userA", "userB", "userC"])
  .return_to_app
Data Collections

• Ancile supports a *Collection* class that stores multiple data values with individual policies
• Policy on the collection is synthesized as the intersection of the policies associated with the data values in the collection
• Allows us to support transformations on collections
  • Map, reduce, and filter
• Example: Training a machine learning location predictor where you want to filter out all non-mobile locations before training
Ancile Performance

• Latency of Ancile requests vary greatly
  • Length of executed program
  • Latency of data providers

• Focus on performance of policy evaluation
  • Depends on the complexity of the policy
  • Between 1 and 15 microseconds for evaluation

• Overhead for compiling program and fetching user credentials
  • 30 to 90 milliseconds for our example programs
  • Caching this info reduces overhead to 3 to 9 milliseconds
Ancile for Research and Enterprise

• We’re looking for customers to use our system

• IRB Compliance
  • Users are more willing to participate with strong privacy guards in place

• Enterprise data privacy compliance
  • Enforces internal privacy guidelines
Conclusion

• Identified privacy challenges that arise in ubiquitous computing applications

• Showed that use-based privacy can be leveraged in location based ubiquitous computing applications

• Demonstrated that Ancile can act as a privacy preserving run-time monitor with minimal overhead
Questions?

Please take our survey

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE
Trusted CI at PEARC19
July 28 - August 1, Chicago

https://www.pearc19.pearc.org/
NSF Cybersecurity Summit
Oct. 15 - 17 in San Diego

Registration and CFP are open!

https://trustedci.org/summit
About the Trusted CI Webinar series

To view presentations, join the announcements mailing list, or submit requests to present, visit: https://trustedci.org/webinars

The next webinar is August 26th at 11am Eastern.
Topic: Scientific Workflow Integrity with Pegasus (SWIP)
Speakers: Anirban Mandal and colleagues