Research Update

Wesley Nuzzo

April 23, 2023

1 progress on the paper

I went through some examples of the typechecking rules for declassification and assignment work, based on the previous examples we used for Noninterference according to policy. I'll show those examples here.

I also reviewed the rules from the wikipedia article to get better idea of how the basic rules work, especially the program counter. I'm not copying those here, though, to save time.

1.1 Declassification and Assignment Examples

I've modified the examples a little bit to suit the model. In particular, instead of guarding the assignment to y with an if statement (which is sufficient for noninterference according to policy), I use the declassify statement.

For all examples, assume $\Gamma(x) = \texttt{secret}$, $\Gamma(y) = \texttt{public}$, and $\Gamma(z) = \texttt{secret} \searrow^c \texttt{public}$. We also asume $\Gamma(c) = \texttt{secret}$.

For all of these, I'm going to ignore the inference rules involving pc to simplify things.

1.1.1 example 1

Example 1: $x := m$ with $\Gamma(m) = \texttt{secret} \searrow^c \texttt{public}$ passes typechecking.

$$\dfrac{\dfrac{\vdash \Gamma(m) \leq \texttt{secret} \searrow^c \texttt{public}}{\Gamma \vdash m : \texttt{secret} \searrow^c \texttt{public exp}}\ \text{T-Var} \qquad \vdash \texttt{secret} \searrow^c \texttt{public} \leq \Gamma(x)}{pc, \Gamma \vdash x := m\ \ \texttt{com}}\ \text{T-Assign}$$

1.1.2 example 2

$y := m$ with $\Gamma(m) = \texttt{secret} \searrow^c \texttt{public}$ fails typechecking.

$$\dfrac{\Gamma \vdash m : \texttt{secret} \searrow^c \texttt{public} \qquad \nvdash \texttt{secret} \searrow^c \texttt{public} \leq \Gamma(y)}{pc, \Gamma \nvdash y := m\ \ \texttt{com}}$$

1.1.3 example 3

$y := \textbf{declassify}(m, \texttt{secret} \searrow^c \texttt{public}\ \textbf{to}\ \texttt{public}, \textbf{using}\ c)$ with $\Gamma(m) = \texttt{secret} \searrow^c \texttt{public}$ passes typechecking.

$$\frac{\Gamma \vdash m : \texttt{secret} \searrow^c \texttt{public} \qquad \vdash \texttt{public} \leq \Gamma(y) \qquad \dfrac{\vdash \Gamma(c) \leq \Gamma(y)}{\Gamma \vdash c : \Gamma(y)} \qquad c \vdash \texttt{secret} \searrow^c \texttt{public} \leq \texttt{public}}{\texttt{public}, \Gamma \vdash y := \textbf{declassify}(m, \texttt{secret} \searrow^c \texttt{public}\ \textbf{to}\ \texttt{public}, \textbf{using}\ c)\ \texttt{com}}$$

The first two conditions are similar to the previous example, except that in the second we use `public`, because that's the level we're declassifying to.

The condition $\Gamma \vdash c : \Gamma(y)$ requires that condition c be at least as public as y. In this case, both are `public`, so that passes.

The condition $c \vdash \texttt{secret} \searrow^c \texttt{public} \leq \texttt{public}$ shows that it is safe to declassify the policy under condition c. In this case, it passes following the rule RL-Decl from the relabeling judgements.

2 Ideas for application to real-world model

I had three main ideas here.

In all cases I'm envisioning an online webserver where data is collected, and users have the ability to submit requests for their data to be deleted through this server.

Of these, my main focus is on the mailing list idea.

2.1 idea 1: mailing list

Users sign up to the mailing list by entering a login and an email address. The server can then send emails to all users on the mailing list using the addresses it has.

Users can use their login to update their email address or delete it from the server entirely. Policy should enforce erasure of the original email address when it gets updated, and erasure of any email address that gets deleted, along with account info.

2.2 idea 2: fitness tracker

A simple fitness app that records things like number of steps taken per day, and so on. Data is grouped by date and time, and users would have the ability to see their own data or delete it for a particular category or time period.

Could maybe add the option to declassify data to send it to, e.g. a doctor or a fitness coach.

2.3 idea 3: ad service

A server that stores data on a users demographics and interests, and can serve ads based on that information.

Users would be able to query the database to see what the database categorizes them as, and to request the deletion of some or all of that data.

2.4 further ideas to consider

A couple other thoughts:

It might be worth adding a way to show/send the user a confirmation of what data has been deleted. It also might be worth thinking of how to prove to the user that the erasure condition was set.

It may be worth looking into how to make it possible to make backups of the data on the system without violating the erasure conditions when a backup is restored. My thinking is that erasure policies could be converted to declassification policies, where the data is only declassified to its original erasure policy after confirmation has been received from the user that the data is still usable.