# Information Flow Control and Privacy

Dennis Chen
Dennis.Chen@tufts.edu
Ming Chow

## Abstract

With the rise of technology and the worry of protecting private user information, a new security practice must be developed. More data is being stored online than ever before and it needs to be protected. Personal information must also be safe and should not fall into malicious hands. Common ways of securing data include firewalls, cryptography and using public/private keys, but nothing prevents information from propagating on. One way to inhibit propagation is to use information flow control. This process uses a type system and labels to secure data from being accessed by malicious parties. With more personal data being stored on computers, people need an easy and effective way of securing their privacy. Information flow control (IFC) is a possible solution to this because of the fact that it can track outputs through a program and prevent leaks. In this day in age where privacy is becoming more important, protecting data is a top priority.

## 1.0 Introduction

As the Internet and technology continue to make their way into households and everyday life, there are many tasks such as online shopping, mail, and banking that are becoming more digitized. The number of users on the Internet has increased from 361 million users (5.8%) in 2000 to 2,749 million users (38.8%) in 2013 [1] and it will continue to grow. With more people using smartphones, computers and other devices, the threat of malicious parties stealing personal information also increases. There has always been a concern about protecting private user information, but there are no security measures in place to counter the rise in technology. One main example of this is smart phone applications and how they may leak private user information without consent to outside parties.

Information flow control (IFC) is a developing concept where a system can monitor the flow of information from one place to another and prevent the flow if it is not wanted. It is a security measure that monitors information propagation between a system and the world, otherwise known as the Internet [2]. Users want to keep their credentials confidential and so IFC uses type-systems and enforces this through compile-time type checking [3]. The basic model has data assigned a security label of confidential (high) or public (low) where the system will make sure no high data flows to a low context. If IFC can be implemented into existing and future phone/web applications, guaranteeing protection of user privacy would be much easier.

## 2.0 Why care about the growing use of technology?

### 2.1 Privacy

The amount of users on the Internet has grown significantly and will continue as activities take advantage the number of people on app stores and websites. In 2012, mobile traffic in the world amounted to 5 exabytes and it is expected to grow to 21 exabytes by 2017 [4].

With more phones in people's pockets, the demand and supply for phone applications also increases. Yet there are pros and cons to this progress. On one hand, apps will be developed and they will accomplish many different tasks such as finding the weather, taking pictures, checking mail, etc. Having all of these make life easier for users because it consolidates everything they need to do into one portable device. The disadvantage to this comes from the fact of what mobile phones are capable of. Phones store contacts and other private information that may be leaked with the use of published applications [5]. For example, a weather app may ask to use the user's location through GPS and then pass it on to outside vendors so that they can target advertisements based on where they are. Applications ask for permissions before users install them and so users can figure out whether they want to trust the application or not. At first this seems like a good system, but once permission has been granted, the user has no way of controlling what the application can do with that information [5].

The current ways of protecting confidential data are typically access control, firewalls, cryptography, and using keys, but none of these are sufficient enough to properly guard against data leaks. Access control works well because it requires privilege in order to access confidential files, but it has no way of verifying propagation [3]. If an application gets ahold of private data because it has access to it, the program can then send this data to outside sources that should not have privilege. Firewalls work through the prevention of communication with other systems. At first this seems like the best option, however it does not provide end-to-end security because practical firewalls permit communication that can violate confidentiality [3]. They do this or else no information would be able to leave and most systems would be rendered useless. Encryption is similar to access control in that once the data is decrypted; the receiver may not follow the confidentiality policy because there is no way to enforce it [3]. This is where information flow control and information flow policy come into play.

## 2.2 Information Flow Control

IFC trumps the other practices by enforcing security policies through tracking information as it propagates through different systems. Research has been done on flow control, but it needs more exposure. The flow of data is often analyzed by type checking, which is the more common approach for dealing with confidentiality. Analysis using type-checking works by incorporating two parts into a security variable: an ordinary type and a security label that dictates how the variable can be used [3]. The type-checker is static and so the compiler reads the program with these labels and makes sure that there are no errors in the information flows. Figure 1 shows an example of a simple program that uses labels.

```
int A@high = 10;
int B@low;
B = A;
```

**Figure 1: This is an example of a program that would not work. A variable B in a low context cannot be assigned a value that is marked in a high context.**

Static analysis is helpful for error detection during compile-time, but some may argue that security cannot be enforced purely statically [6]. Programs typically interact with outside systems and these interactions cannot be predicted beforehand. An example of this is when security settings need to be changed in a file/database. To counter that, there are also dynamic techniques that enforce security policies during runtime by terminating the program when a possible leak is about to happen [2]. The problem with dynamic information flow control is that it has a large run-time overhead and that it cannot prevent side-channel attacks. However both of these problems are solved through static information flow control and so a combination of the two make for the best security practice [6].

## 3.0 Applications of IFC

### 3.1 Protecting Private Information

Information flow control can deal with a few major problems that users have in today's society. One concern that is often heard on the news is the worry of big companies harvesting data and selling it to third parties for separate use. Companies like Facebook and Google have access to user data such as birthdays, search patterns, geographic locations, and much more. Even though most users do not want these companies to give out private information, they do not want to stop using these services because of how popular/efficient they are. IFC would be the perfect solution to this since it allows users to give access to specific entities and prevent them from sharing that information. Looking at Figure 2, Alice has a secret message that she wants to send to Bob and does not want him to pass that information along to anyone else. Alice can label that message with Bob's permission (and her own) and that makes it so that only those two can read it. In addition to companies selling private information, there are more malicious things that could be done. A start to a solution would be to incorporate labels and type systems into existing code.

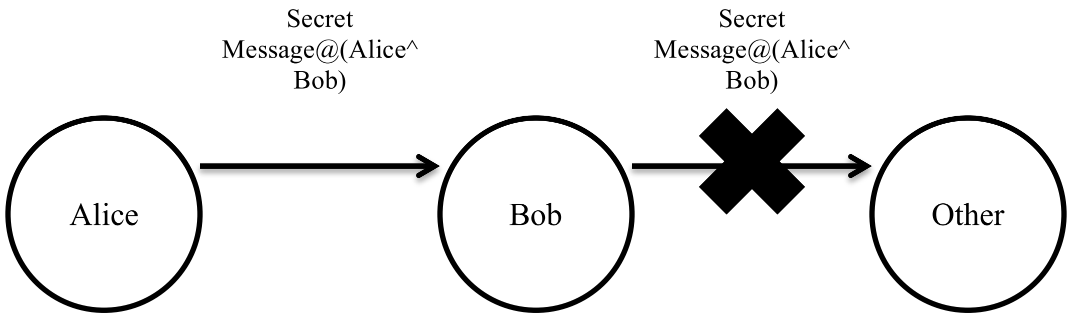

**Figure 2: Alice sends Bob a message that is labeled only for the two of them. When Bob tries sending that message to Other, the type system does not let him. Thus preventing a leak in formation**

## 3.2 Protecting Against Covert Channels

Another problem that can be solved by IFC is the prevention of covert channels. Protecting user credentials and other actual data/information is usually a top priority in computer security, but there is other information that can reveal secrets as well. A covert channel is an attack against a system that involves leaking information whose primary purpose is not information transfer [3]. There are many different kinds of covert channel attacks such as implicit flows, timing channels, resource exhaustion channels, and power channels [3]. Power channels have information about the power consumed by the computer, resource exhaustion channels provide numbers on memory/disk limitations, and implicit flows can leak structural information of a program [2]. Covert channels can be just as dangerous as normal channels because attacks can observe these channels and gain knowledge that way. For example, an attacker could make use of timing channels by obtaining the total execution time of a program depending on its input. This means that it is possible for them to test input and get a sense of how the underlying program is written.

## 3.3 Making privacy easier to use

One big problem with security and privacy originates from users and the usability of security software [7]. When people are on their computers or phones, they are generally on them for a reason; they want to send emails, browse web pages or download software. Security is usually put as an afterthought since it is not the main goal. Anything that prevents or hinders users from accomplishing their task is a hindrance. Security is another layer that must be done before completing their task. Integrating IFC as a security practice could protect users better by actually implementing it in application code.

Going off of this, a possible solution to protecting private information comes from research done on user-aware privacy control. Microsoft did research on how it is possible to incorporate security into devices by building applications on a mobile-platform designed for security [5]. This platform, named TouchDevelop, has developers write apps using an expressive scripting language and allows users to monitor security flow with ease. The approach works mainly through static analysis and it uses IFC to see if any data is leaked. For example, an application may want to use a user's location as a way to figure out what restaurant is closest to them, but the user does not want that information to be published anywhere else. This solution counters the general argument that users cannot be bothered to deal with security because in this case the language itself deals with that. TouchDevelop provides an easy to use interface for users, which has them sliding bars to grant access to information [5]. When writing scripts, the platform is able to compute information flows and judge them accordingly so that the developer does not have to [5]. This simplifies a developer's job because they can then focus more on the app and worry less about implementing security.

## 3.4 IFC Opponents

Although information flow control offers a lot of benefits in terms of security, opponents argue that actually implementing this it would be impractical. As mentioned before, the use of static and dynamic analysis solves the problem of larger scale systems because static IFC manages compile-time while dynamic IFC deals with run-time. However, while dynamically checking, there is a chance of a bad flow and the program halts in order to prevent a leak. IFC violations are not recoverable and these errors terminate the entire system immediately [2]. This protects private information, but having the program stop is not feasible in real world programs because the program loses availability.

The solution to this problem is through the use of not-a-values (NaVs) to propagate an error through the program until it can be terminated safely [2]. Using public labels and delayed exceptions, NaVs allow errors to be recoverable even in IFC violations. This way, NaVs act as a normal variable and propagate through the program if there is a leak of information in the beginning of the program. Then if the NaV is used for something it cannot be used for, it fails and reveals its propagation path so that it can be fixed while other processes are still running. While writing code to recover errors is not easy, it does make error handling possible and worthwhile.

## 4.0 Conclusion

There are many different security mechanisms in the world that are used to protect user private information, but currently there are no practices that allow for end-to-end security. IFC has shown that it can be a good alternative to current practices because of its ability to monitor flows and prevent information from leaking. IFC consists of dynamic and static information flow where both have their own advantages and disadvantages. Static IFC excels at compile-time and reducing overhead costs whereas dynamic IFC is used during run-time for large-scale systems. Combining both of these together provides for a more secure system. Managing privacy is another task that many people want, but do not want to deal with. Fortunately IFC takes care of the bulk of this since would do privacy checking for the users. Currently, exposure of functional programming and IFC is limited. However, with increasing concerns of privacy and more schools teaching functional programming, the use of information flow control will rise.

## 5.0 References

[1] "Internet World Stats: Internet Growth Statistics," August 12, 2013. [Online]. Available: http://www.internetworldstats.com/emarketing.htm.

[2] C. Hritcu, M. Greenberg, B. Karel, B. C. Pierce, and G. Morrisett, "All Your IFCException Are Belong To Us," [Online]. Available: http://www.cis.upenn.edu/~mgree/papers/sp2013_exceptional.pdf.

[3] Andrei Sabelfield and Andrew C. Myers, "Language-Based Information-Flow Security," [Online]. Available: http://www.utd.edu/~hamlen/Papers/sm-jsac03.pdf.

[4] Victor H., "Rise of smartphones and tablets to bring 300% to mobile data by 2017," 3 July 2013. [Online]. Available: http://www.phonearena.com/news/Rise-of-smartphones-and-tablets-to-bring-300-to-mobile-data-by-2017_id44865.

[5] X. Xiao, N. Tillmann, M. Fahndrich, J. de Halleux, and M. Moskal, "User-Aware Privacy Control via Extended Static-Information-Flow Analysis," [Online]. Available: http://research.microsoft.com/pubs/169640/ase2012-privacy.pdf.

[6] Lantian Zheng and Andrew c. Myers, "Dynamic Security Labels and Static Information Flow Control," [Online]. Available: http://www.cs.cornell.edu/andru/papers/dynlabel-ijis.pdf.

[7] Alma Whitten and J. D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," [Online]. Available: http://www.gaudior.net/alma/johnny.pdf.