Measuring the leakage of PII from web sites and apps

Northeastern

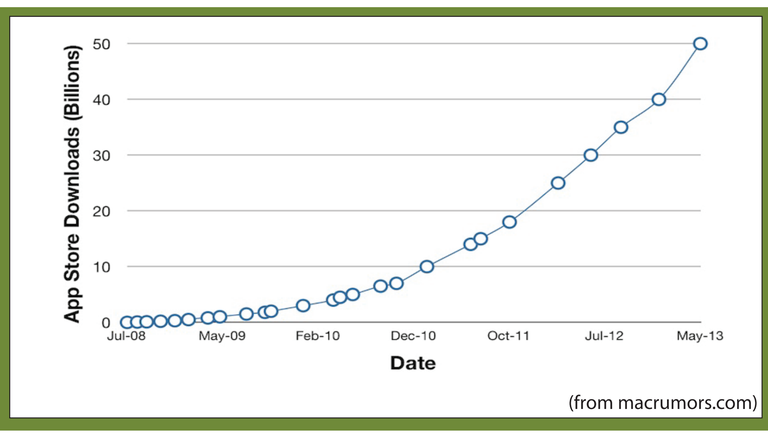
NARUS

facebook skype flickr YouTube Broadcast Yourself

Users commonly provide significant **personal information (PII)** to popular web sites, smartphone apps

Little visibility into what apps do with PII

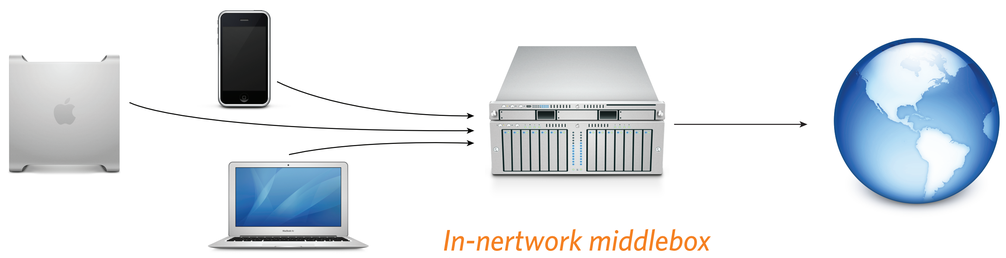
Automatically detect PII leaks



(from macrumors.com)

APPROACH

Network-based approach using **middlebox**
- Observe traffic, locate leaks of information
- Higher coverage than end-host-based approach



But, many **challenges in analyzing data**
- Collection easy, extracting relevant PII difficult

CHALLENGE 1: LOCATING FIELDS

Apps, web sites use many **different protocols**
- PII buried inside parameters, URLs, content itself, ...

HTTP HTTPS SPDY SMTP UDP WebRTC CUSTOM

Initial directions:

1. Use **man-in-the-middle proxy** to view HTTPS

2. Use **protocol parsers** to decontruct fields

```
HTTP
GET /foo.html?name=Alice 1.1
Cookie: a=293&g=f&age=39&id=27
ETag: 2039-2dc90ea2-12
```

```
name -> Alice
a -> 293
g -> f
id -> 27
age -> 39
```

CHALLENGE 2: LOCATING PII

Users today have **many types of PII**
- Need approach to determine what fields contain PII

1. *Passive measurements*
   - Assume **knowledge of some users' PII**
   - Assume visibility into many network flows
   - Correlate fields with known information

KNOWN PII			DATA FROM OBSERVED TRAFFIC			
Name	Gender	Age	a	g	age	id
Alice	Female	39	293	f	39	27
Bob	Male	49	1022	m	48	49
Charlie	Male	24	992	m	24	200
Dave	Male	64	108	m	--	--
Eve	Female	33	221	f	33	1929

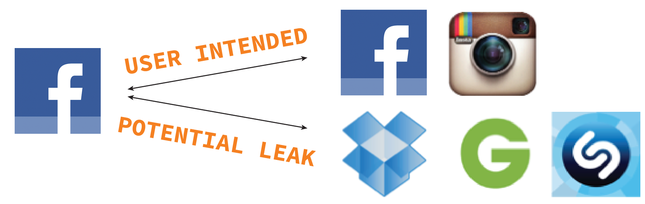
2. *Active measurements*
   - Create **synthetic accounts** and data
   - Use automation to interact with site or app
     - Use MonkeyRunner, testing frameworks
   - Search for synthetic data present in different fields

For **non-recognized fields**, can try to reason about
- Measure entropy, look for common formats, ask user

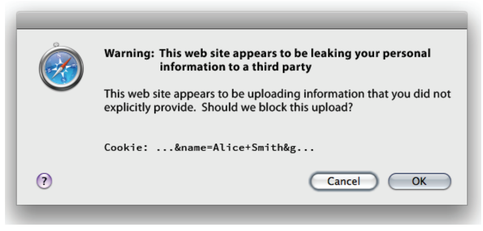
CHALLENGE 3: FINDING LEAKS

Goal: **Identify PII leaks** to first and third parties

Understand entity relationships (e.g. FB owns Instagram)
- Periodically survey users about observed leaks
- Learn leaks commonly desired, not



Eventually: **Inform user** of what is being leaked
- Warn user before leak happens, allow user to stop



Yabing Liu[†] Alan Mislove[†] Ignacio Bermudez[‡] Han Song[‡] Alok Tongaonkar[‡] Mario Baldi[‡] Antonio Nucci[‡]

[†]*Northeastern University* [‡]*Narus*