CSE 484 / CSE M 584: Computer Security and Privacy

Anonymity
Mobile

Autumn 2018

Tadayoshi (Yoshi) Kohno
yoshi@cs.Washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Ada Lerner, John Manferdelli, John Mitchell, Franziska Roesner, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Admin

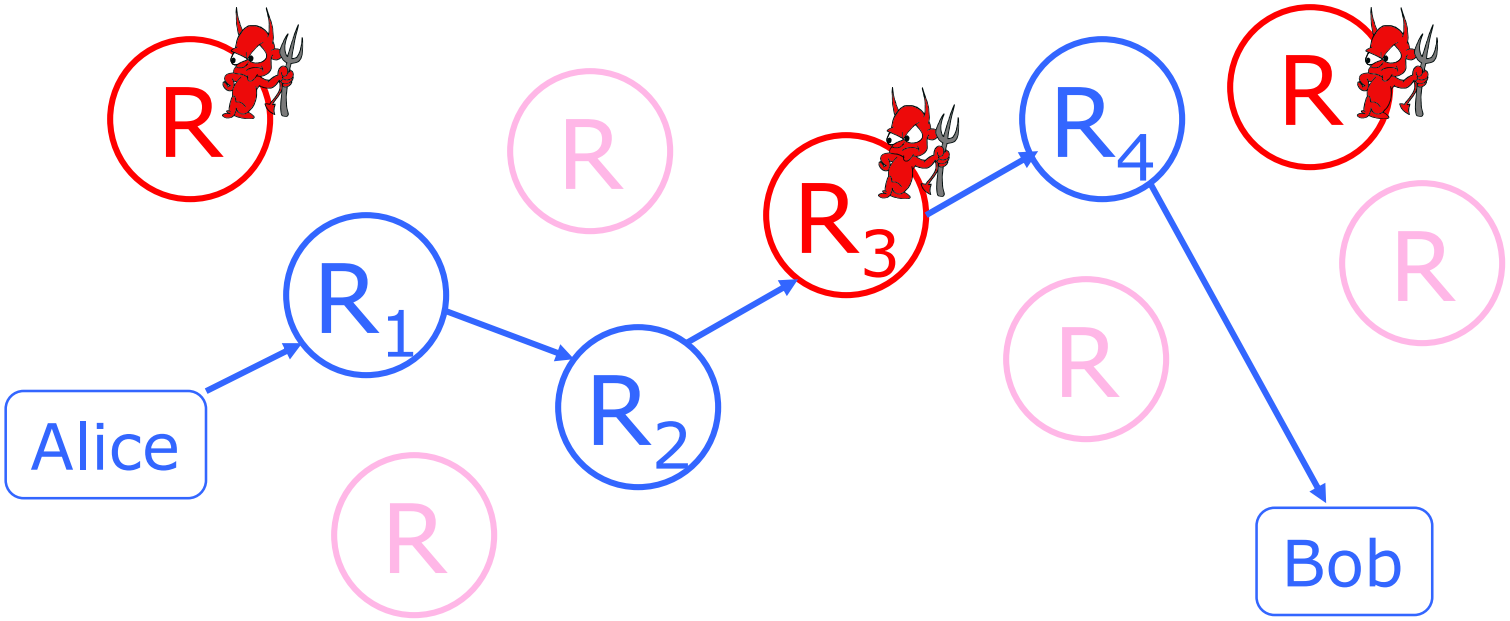
- HW 3 due Nov 30
- Lab 3 out this week, due Dec 7 (Quiz Section on Nov 29)

Admin

- Final Project Proposals: Looked great!
- Final Project Checkpoint: Nov 30 – preliminary outline and references
- Final Project Presentation: Dec 10 – 12-15-minute video – **must** be on time
- Explore something of interest to you, that could hopefully benefit you or your career in some way – technical topics, current events, etc

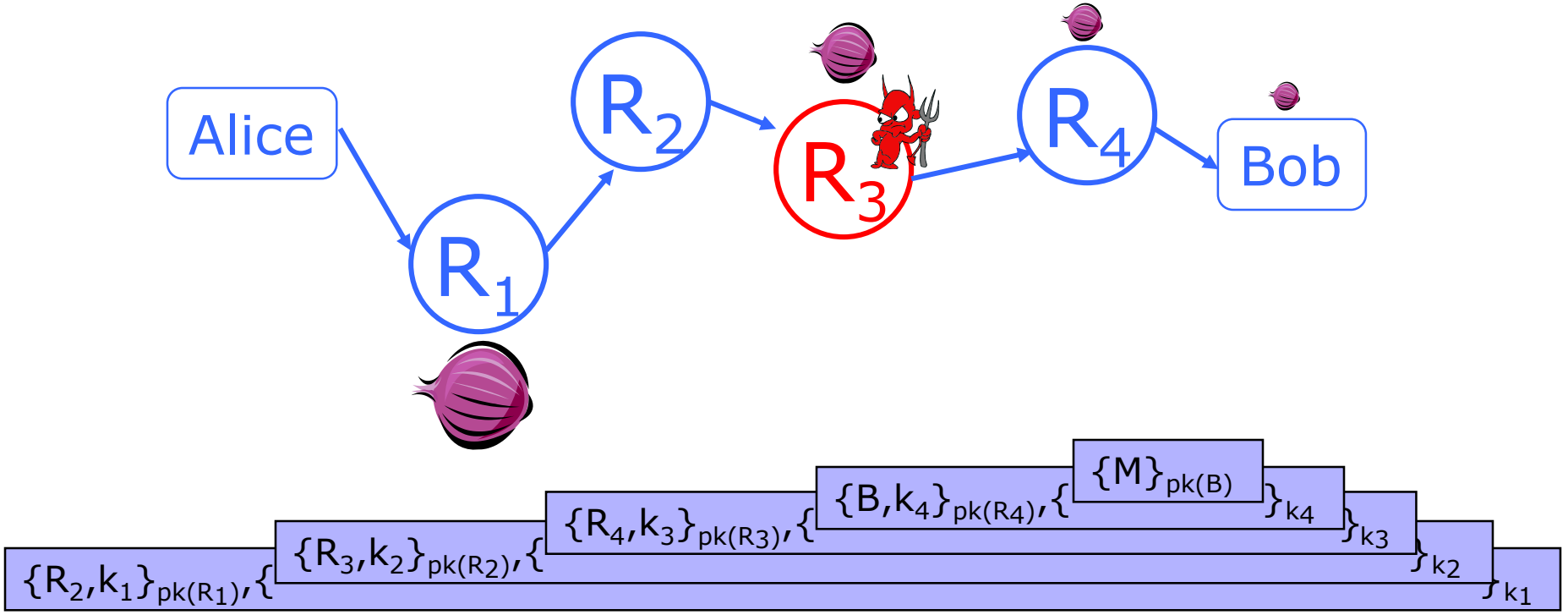
[Reed, Syverson, Goldschlag 1997]

Review: Onion Routing



- Sender chooses a random sequence of routers
  - Some routers are honest, some controlled by attacker
  - Sender controls the length of the path

Review: Route Establishment



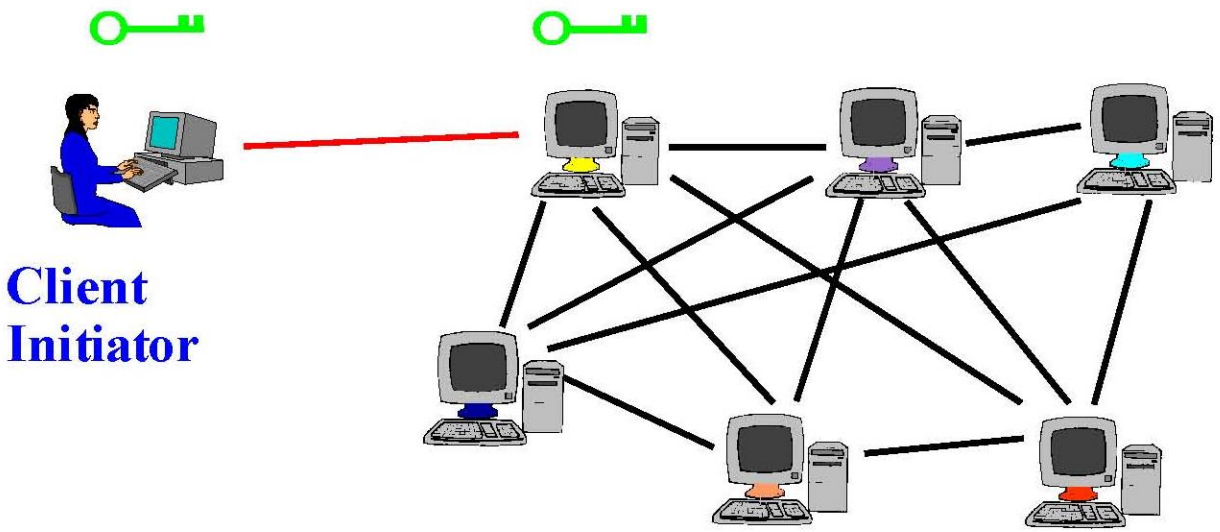
- Routing info for each link encrypted with router's public key
- Each router learns only the identity of the next router

Tor

- Second-generation onion routing network
  - http://tor.eff.org
  - Developed by Roger Dingledine, Nick Mathewson and Paul Syverson
  - Specifically designed for low-latency anonymous Internet communications
- Running since October 2003
- "Easy-to-use" client proxy
  - Freely available, can use it for anonymous browsing

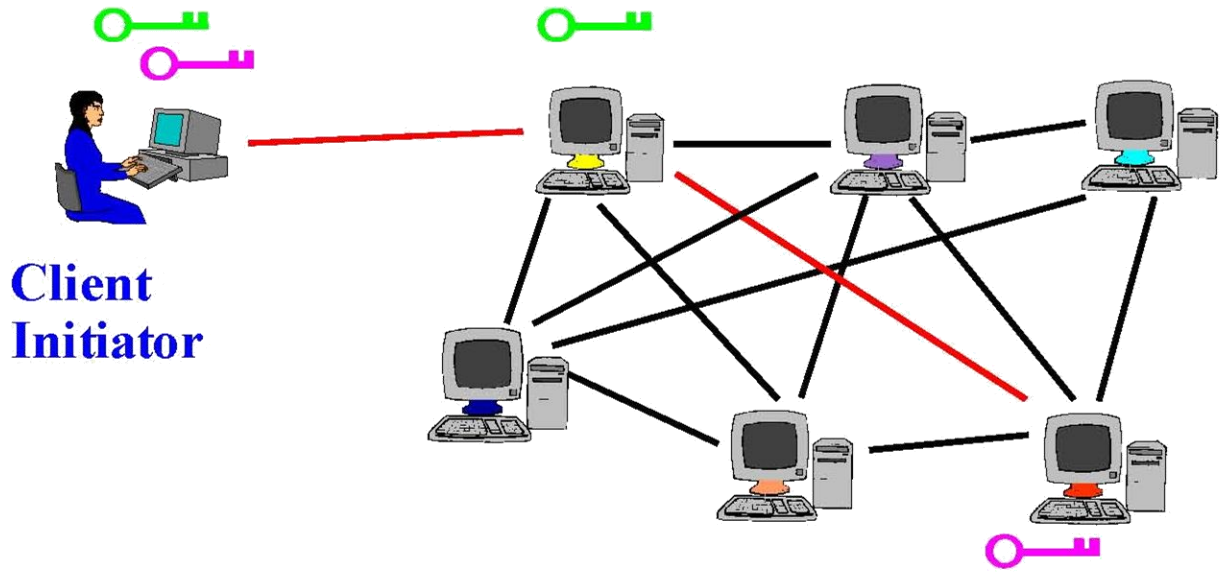
Tor Circuit Setup (1)

- Client proxy establishes a symmetric session key and circuit with Onion Router #1



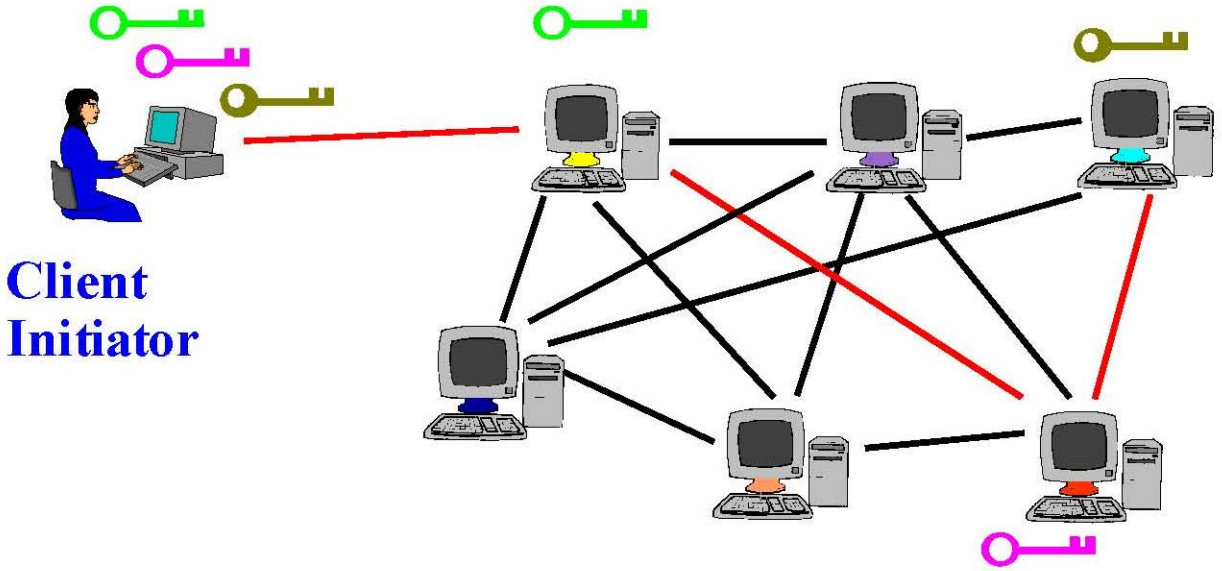
Tor Circuit Setup (2)

- Client proxy extends the circuit by establishing a symmetric session key with Onion Router #2
  - Tunnel through Onion Router #1



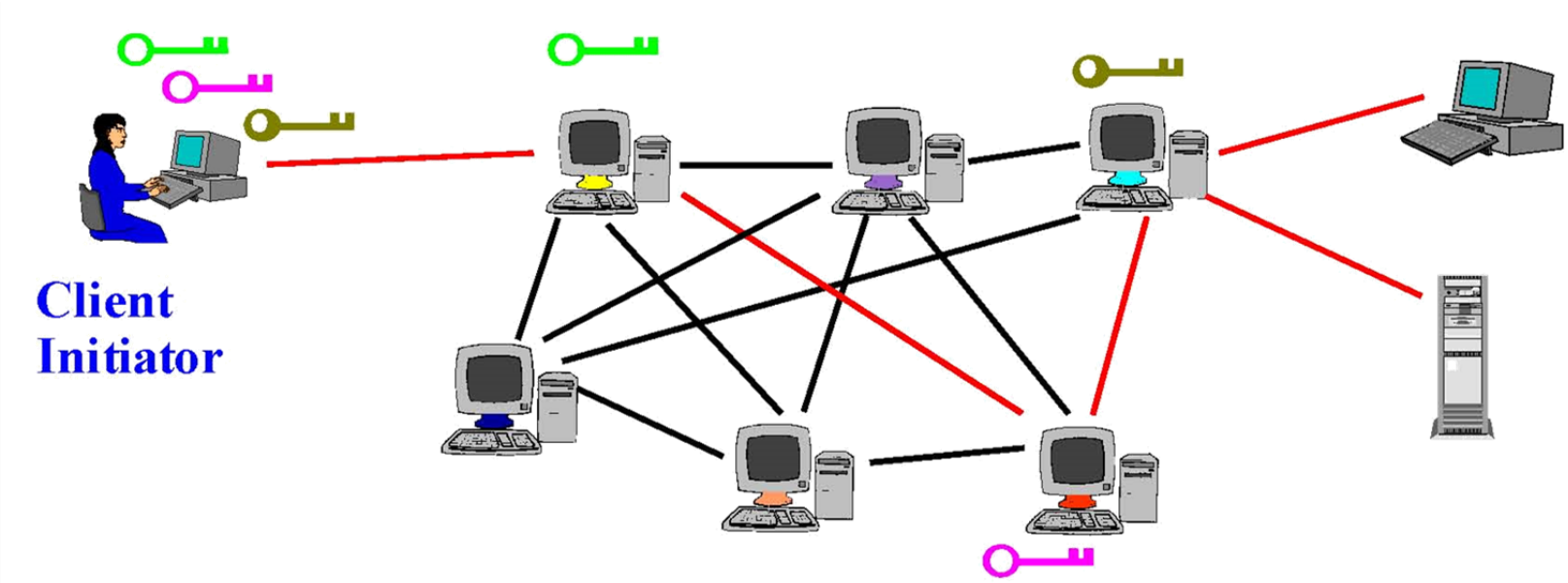
Tor Circuit Setup (3)

- Client proxy extends the circuit by establishing a symmetric session key with Onion Router #3
  - Tunnel through Onion Routers #1 and #2



Using a Tor Circuit

- Client applications connect and communicate over the established Tor circuit.



Tor Management

- Many applications can share one circuit
  - Multiple TCP streams over one anonymous connection
- Tor router doesn't need root privileges
  - Encourages people to set up their own routers
  - More participants = better anonymity for everyone
- Directory servers
  - Maintain lists of active onion routers, their locations, current public keys, etc.
  - Control how new routers join the network
    - "Sybil attack": attacker creates a large number of routers
  - Directory servers' keys ship with Tor code

Is Tor Perfect?

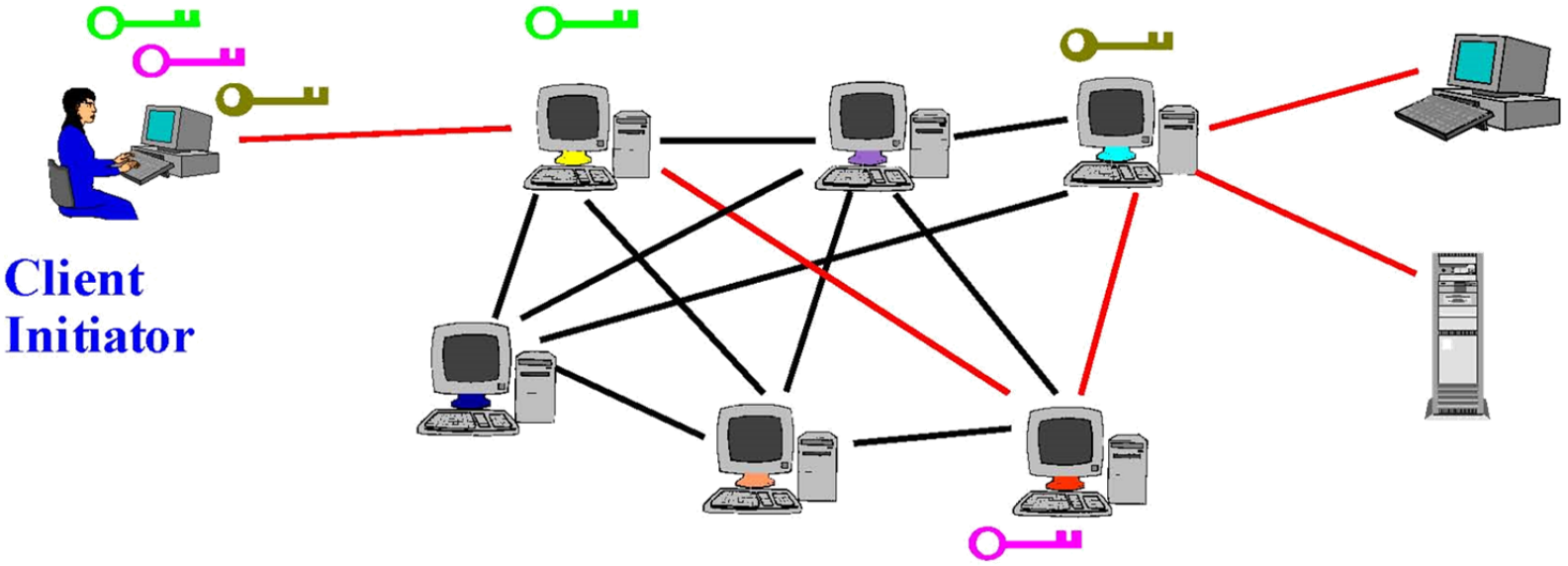
- Q: What can "go wrong" with the use of Tor?

Issues and Notes of Caution

- Passive traffic analysis
  - Infer from network traffic who is talking to whom
  - To hide your traffic, must carry other people's traffic!
- Active traffic analysis
  - Inject packets or put a timing signature on packet flow
- Compromise of network nodes
  - Attacker may compromise some routers
    - And powerful adversaries may have "too many" routers (e.g., a super computer at a national lab)
  - It is not obvious which nodes have been compromised
    - Attacker may be passively logging traffic
  - Better not to trust any individual router
    - Assume that some fraction of routers is good, don't know which

Issues and Notes of Caution

- Tor isn't completely effective by itself
  - Tracking cookies, fingerprinting, etc.
  - Exit nodes can see everything!



Issues and Notes of Caution

- The simple act of using Tor could make one a target for additional surveillance
- Hosting an exit node could result in illegal activity coming from your machine

Mobile Security

Roadmap

- History, How we got here

- Mobile malware
- Mobile platforms vs. traditional platforms
- Dive into Android

Questions: Mobile Malware

Q: How might malware authors get malware onto phones?

Q: What are some goals that mobile device malware authors might have?

Smartphone (In)Security

Users accidentally install malicious applications.

Smartphone (In)Security

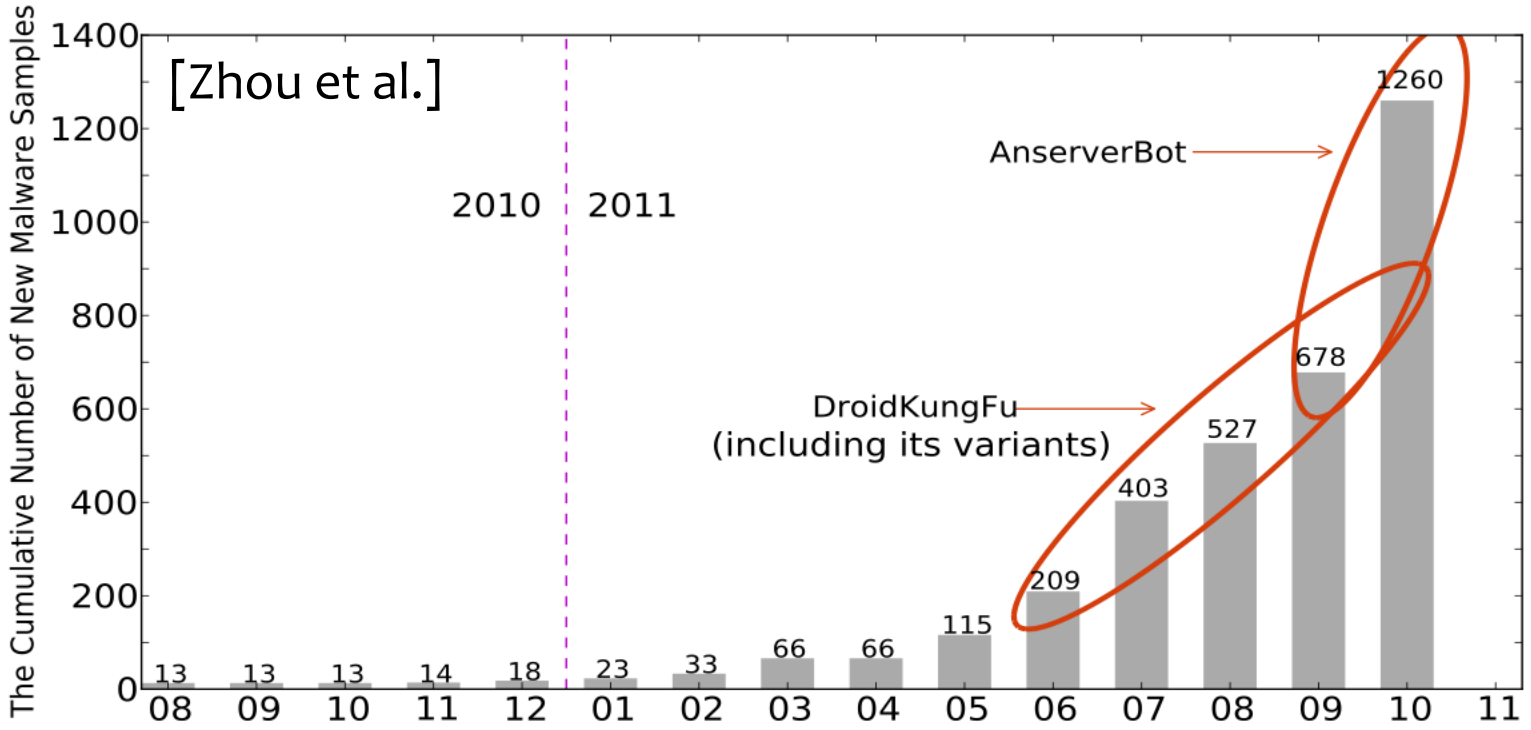
Even legitimate applications exhibit questionable behavior.

Mobile Malware Goals

- "Unique" to phones:
  - Premium SMS messages
  - Identify location
  - Record phone calls
  - Log SMS
- Similar to desktop/PCs:
  - Connects to botmasters
  - Steal data
  - Phishing
  - Malvertising

Malware in the Wild

Android malware grew quickly!
Today: millions of samples.



Mobile Malware Examples Over Time

- **DroidDream** (Android)
  - Over 58 apps uploaded to Google app market
  - Conducts data theft; send credentials to attackers
- **Zitmo** (Symbian,BlackBerry,Windows,Android)
  - Poses as mobile banking application
  - Captures info from SMS – steal banking 2nd factors
  - Works with Zeus botnet
- **Ikee** (iOS)
  - Worm capabilities (targeted default ssh password)
  - Worked only on jailbroken phones with ssh installed

Background: Before Mobile Platforms

Assumptions in traditional OS (e.g., Unix) design:

1. There may be multiple users who don't trust each other.
2. Once an application is installed, it's (more or less) trusted.

Background: Before Mobile Platforms

Assumptions in traditional OS (e.g., Unix) design:

1. **There may be multiple users who don't trust each other.**
2. Once an application is installed, it's (more or less) trusted.

Background: Before Mobile Platforms

Assumptions in traditional OS (e.g., Unix) design:

1. There may be multiple users who don't trust each other.
2. **Once an application is installed, it's (more or less) trusted.**

Apps can do anything the UID they're running under can do.

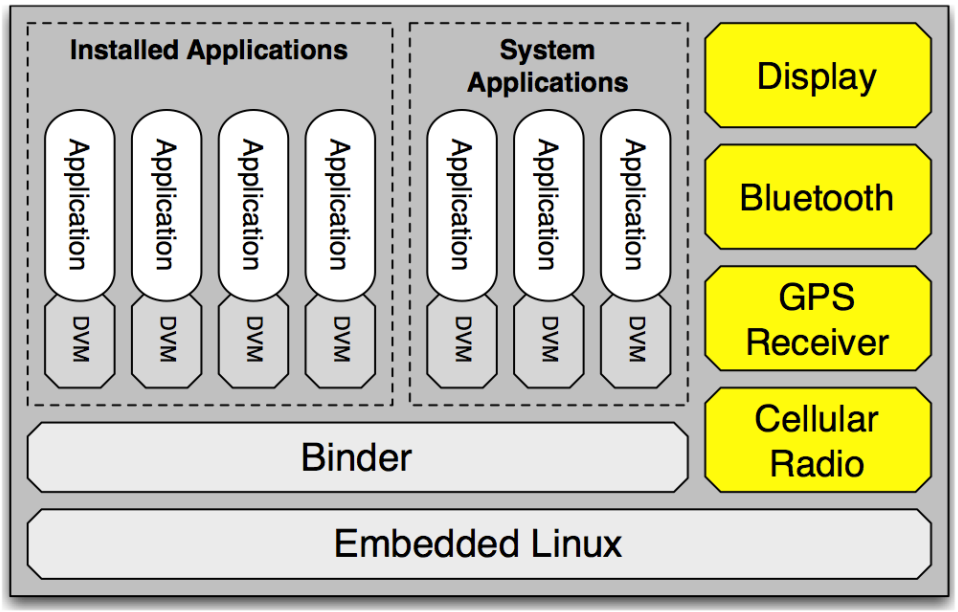
What's Different about Mobile Platforms?

- **Isolation:** Applications are isolated
  - Each runs in a separate execution context
  - No default access to file system, devices, etc.
  - **Different than traditional OSes** where multiple applications run with the same user permissions!

- **App Store:** Approval process for applications
  - Market: Vendor controlled/Open
  - App signing: Vendor-issued/self-signed
  - User approval of permissions

More Details: Android

[Enck et al.]



- Based on Linux
- Application sandboxes
  - Applications run as separate UIDs, in separate processes.
  - Memory corruption errors only lead to arbitrary code execution in the context of the **particular** application, not complete system compromise!
  - (Can still escape sandbox – but must compromise Linux kernel to do so.) ← allows rooting

Challenges with Isolated Apps

So mobile platforms isolate applications for security, but…

1. Permissions: How can applications access sensitive resources?
2. Communication: How can applications communicate with each other?

Permission Granting Problem

Smartphones (and other modern OSes) try to prevent such attacks by limiting applications' access to:

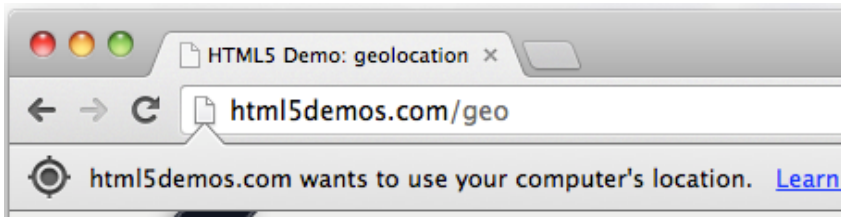
- System Resources (clipboard, file system).
- Devices (camera, GPS, phone, …).

How should operating system grant permissions to applications?

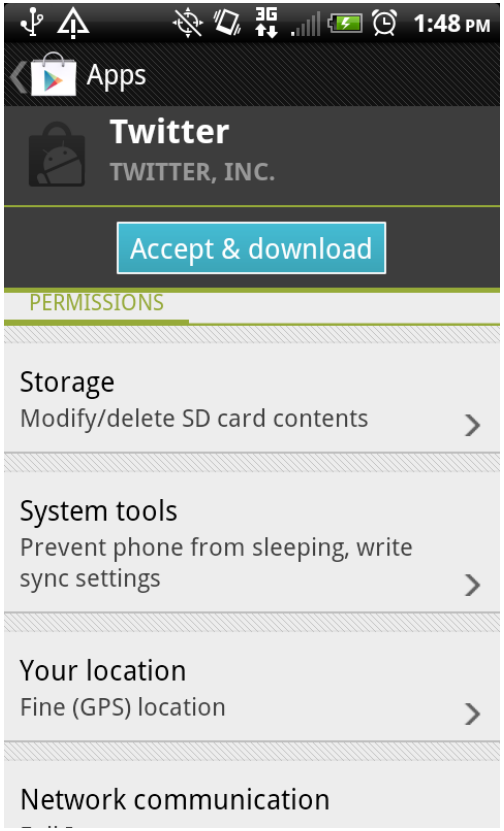
Standard approach: Ask the user.

Two Ways to Ask the User

Prompts (time-of-use)



Manifests (install-time)

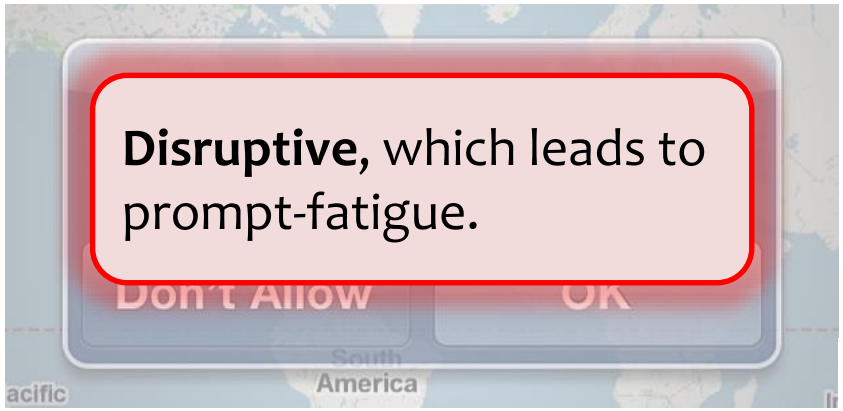


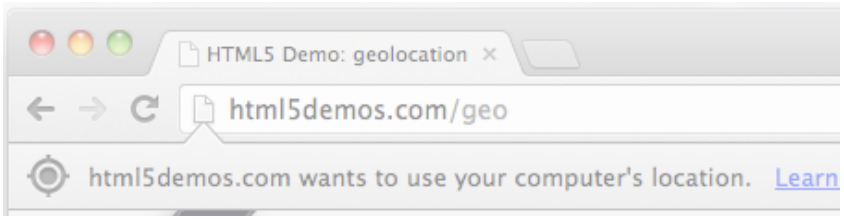
Questions

- Q: What are the pros and cons of the manifest-based permission model?
- Q: What are the pros and cons of the “ask each use” permission mode?

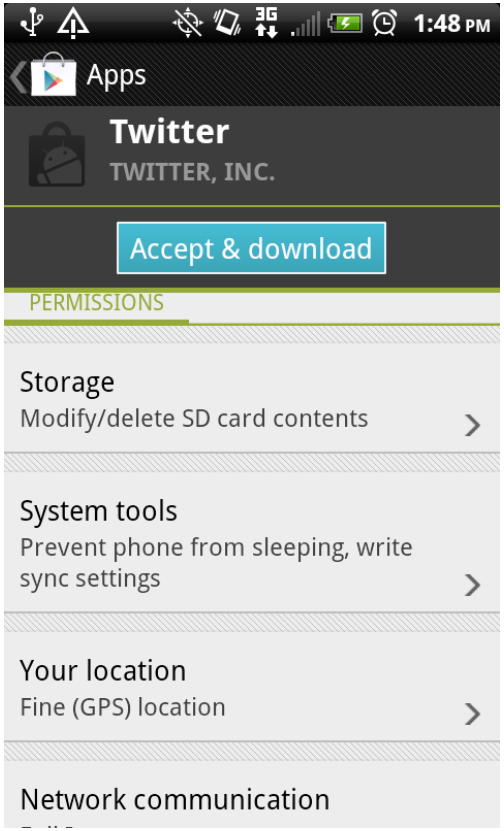
Two Ways to Ask the User

Prompts (time-of-use)



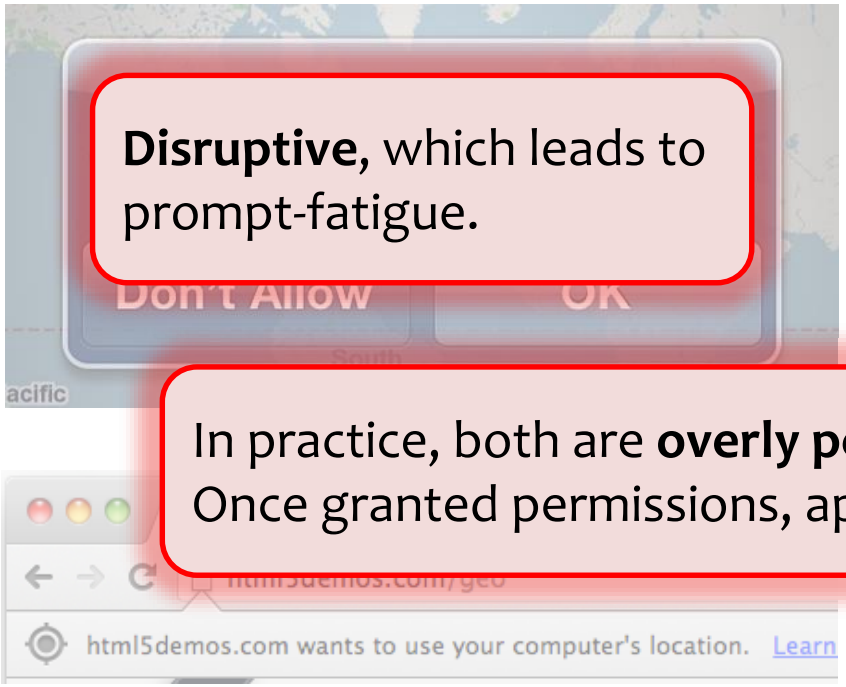


Manifests (install-time)



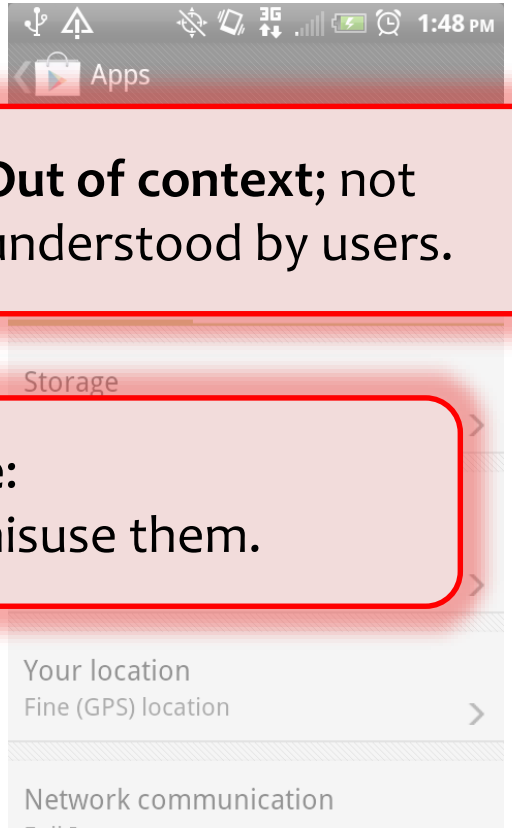
Two Ways to Ask the User

Prompts (time-of-use)



Disruptive, which leads to prompt-fatigue.

Manifests (install-time)



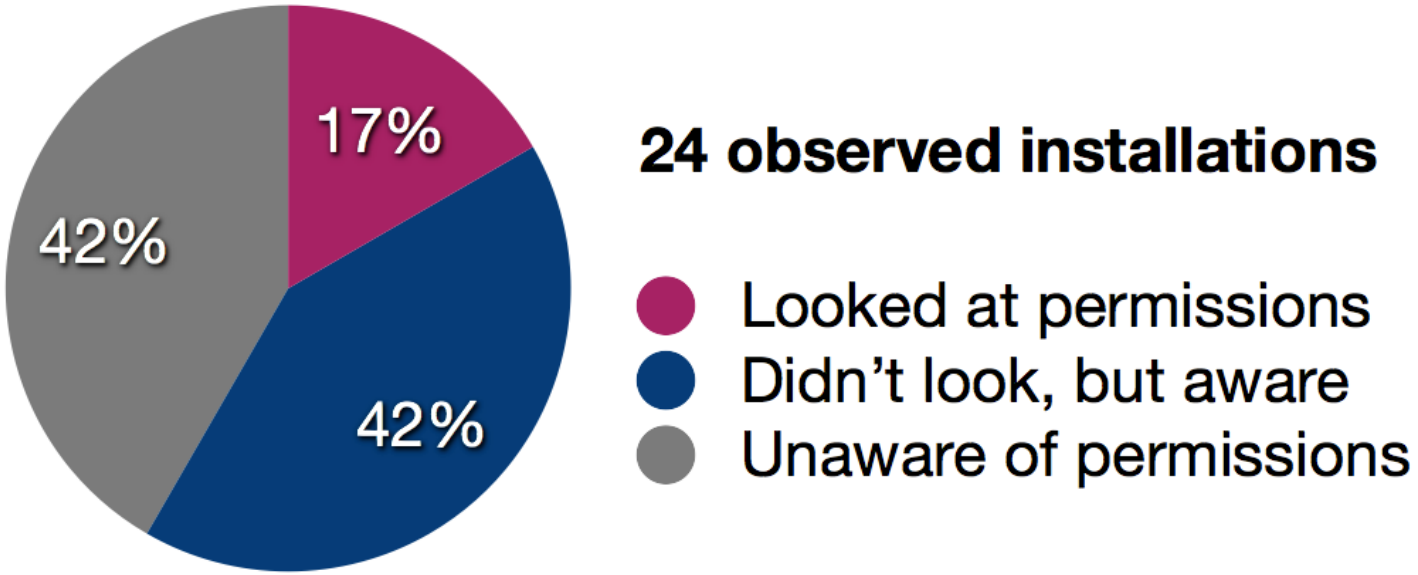
Out of context; not understood by users.

In practice, both are **overly permissive**:
Once granted permissions, apps can misuse them.

[Felt et al.]

Are Manifests Usable?

Do users pay attention to permissions?



… but 88% of users looked at reviews.

[Felt et al.]

Are Manifests Usable?

Do users understand the warnings?

	Permission	n	Correct Answers	
1 Choice	READ_CALENDAR	101	46	45.5%
	CHANGE_NETWORK_STATE	66	26	39.4%
	READ_SMS$_1$	77	24	31.2%
	CALL_PHONE	83	16	19.3%
2 Choices	WAKE_LOCK	81	27	33.3%
	WRITE_EXTERNAL_STORAGE	92	14	15.2%
	READ_CONTACTS	86	11	12.8%
	INTERNET	109	12	11.0%
	READ_PHONE_STATE	85	4	4.7%
	READ_SMS$_2$	54	12	22.2%
4	CAMERA	72	7	9.7%

Table 4: The number of people who correctly answered a question. Questions are grouped by the number of correct choices. n is the number of respondents. (Internet Survey, $n = 302$)

[Felt et al.]

Are Manifests Usable?

Do users act on permission information?

"Have you ever not installed an app because of permissions?"