CE879 - Information Security Mng. & Eng.

Lecture 5: Humans and usable security

Seyedeh Atefeh Musavi / Mehdi Kharrazi
Department of Computer Engineering
Sharif University of Technology
Spring 1404

S4Lab

Acknowledgments: Some of the slides are fully or partially obtained from other sources. A reference is noted on the bottom of each slide to acknowledge the full slide or partial slide content.

THERE'S AN OLD JOKE THAT COMPUTERS ARE ACTUALLY EASY MACHINES TO SECURE: just turn them off, lock them in a metal-lined room, and throw away the key. What you end up with is a machine that is very secure, just not very usable.

[Cranor, L. F., Garfinkel, S., *Security and usability: designing secure systems that people can use,* O'Reilly Media, 2005]

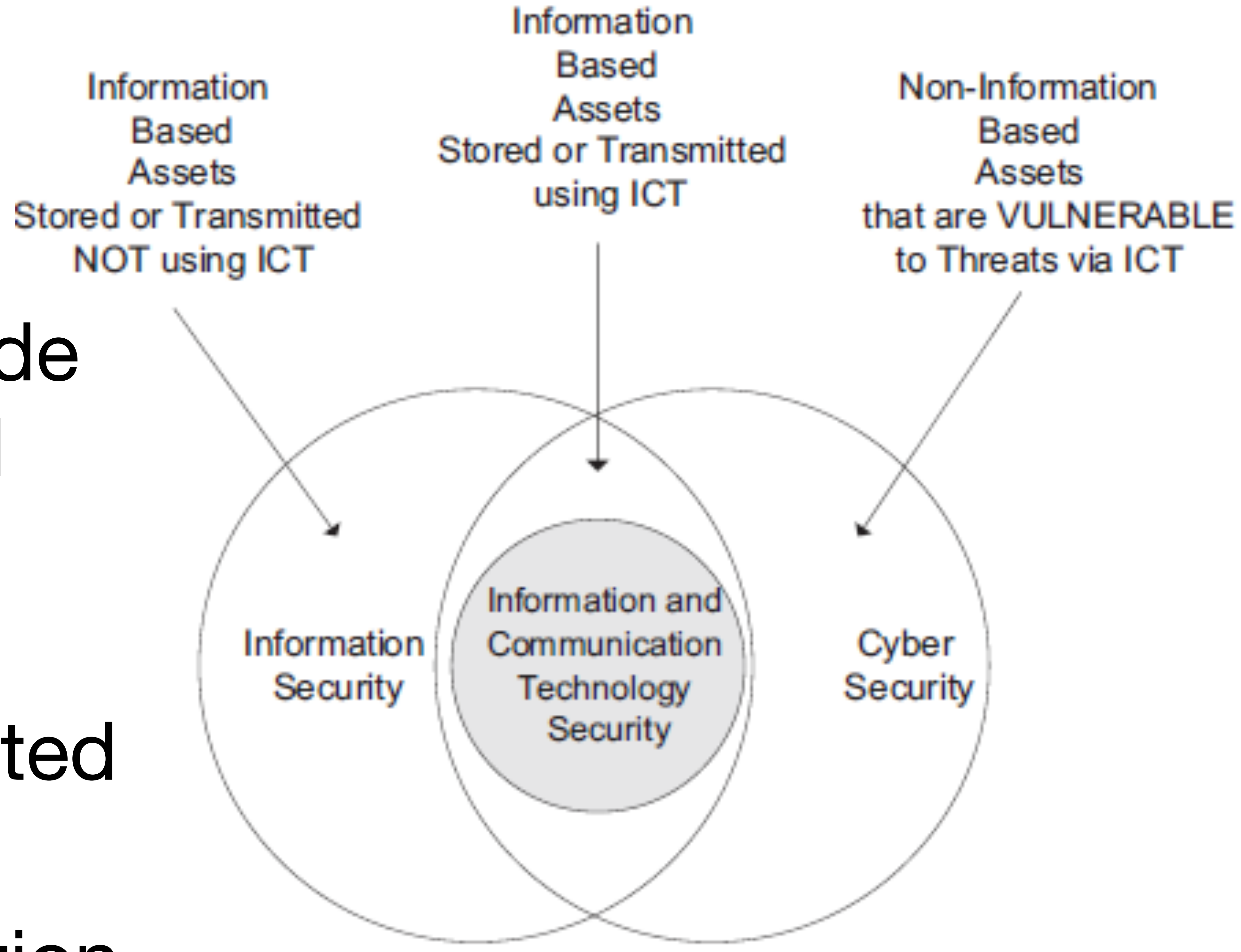
What are the human factors in cyber security?

and why are they important?

From information security to cyber security

S4Lab

- Where are the human factors?
- Remember the three pillars of cybersecurity?
  - Technology, processes, and people.
- The assets cyber security aims to protect include an additional dimension which extends beyond the formal boundaries of information security.
- Both humans in their personal capacity and society at large can be directly harmed or affected by cyber security attacks
  - This is not necessarily the case with information security where harm is always indirect.



[Von Solms, R., & Van Niekerk, J., From information security to cyber security. computers & security, 2013]

Human factors in cyber security

S4Lab

- Humans are consistently referred to as the weakest link in security.
- Human factors influence how individuals interact with security technology.
- Dynamic and complex! Many factors:
  - Influence of individual differences
  - Personality traits
  - Cognitive abilities
  - Biases and heuristics that affect how individuals perceive risk
- Important because they help explain why individuals make certain decisions and why specific behaviors may be observed.

Parsons, K., et al., *Human factors and information security: individual, culture and security environment*. Def. Sci. and tech org, Australia, 2010]

Jeep Shifter Example

S4Lab

- Jeep's shift level doesn't mechanically control the transmission, even though it looks and moves like a traditional shift lever.
  - Fundamentally a software switch that controls the transmission electronically.
- The "Monostable" design doesn't provide any meaningful feedback about what gear you're in — it returns to the center position after each shift.
- LEDs on the shifter (often covered by your palm) or the digital display in the instrument cluster displays current position.
- Confusion for thousands of people!
  - Over a hundred injuries, and now potentially a death.
  - Because of a design that prioritizes screens over switches.

[https://www.theverge.com/2016/6/27/12043898/chrysler-jeep-dodge-electronic-gear-shift-recall-design-flaw-video]

Human Factor Errors

S4Lab

- Reasons for information security breaches:
  - Acts of omission, in which people forget to perform a necessary action.
    - For instance, the failure to regularly change passwords.
  - Errors are commonly acts of commission, in which people perform an incorrect procedure or action,
    - i.e. writing down a password.
  - Extraneous acts, which involves doing something unnecessary.
  - Sequential acts, which involve doing something in the wrong order.
  - Time errors, caused by people failing to perform a task within the required time.

Parsons, K., et al., *Human factors and information security: individual, culture and security environment*. Def. Sci. and tech org, Australia, 2010]

Why users fail to show the required behavior?

S4Lab

- Impossible Demands
  - Most users today find it impossible to comply with standard policies governing the use of computer passwords.
- Awkward Behaviors
  - User locks the screen of his computer every time he leaves the office, even for brief periods.
  - His colleagues likely suspect that the user either has something to hide or does not trust them.

[Cranor, L. F., Garfinkel, S., *Security and usability: designing secure systems that people can use,* O'Reilly Media, 2005]

Why users fail to show the required behavior? (con’t)

S4Lab

- Beyond the User Interface
  - Why Johnny Can't Encrypt.
  - Users' perception of the task of encrypting email vs. the way that the PGP interface presents those tasks to users.
  - User-centered design of security mechanisms, however, is more than user interface design.
  - A cryptographic key does not function like a key in the physical world.
    - People's understanding of "public" and "private" is different from how these terms are applied to public and private keys.

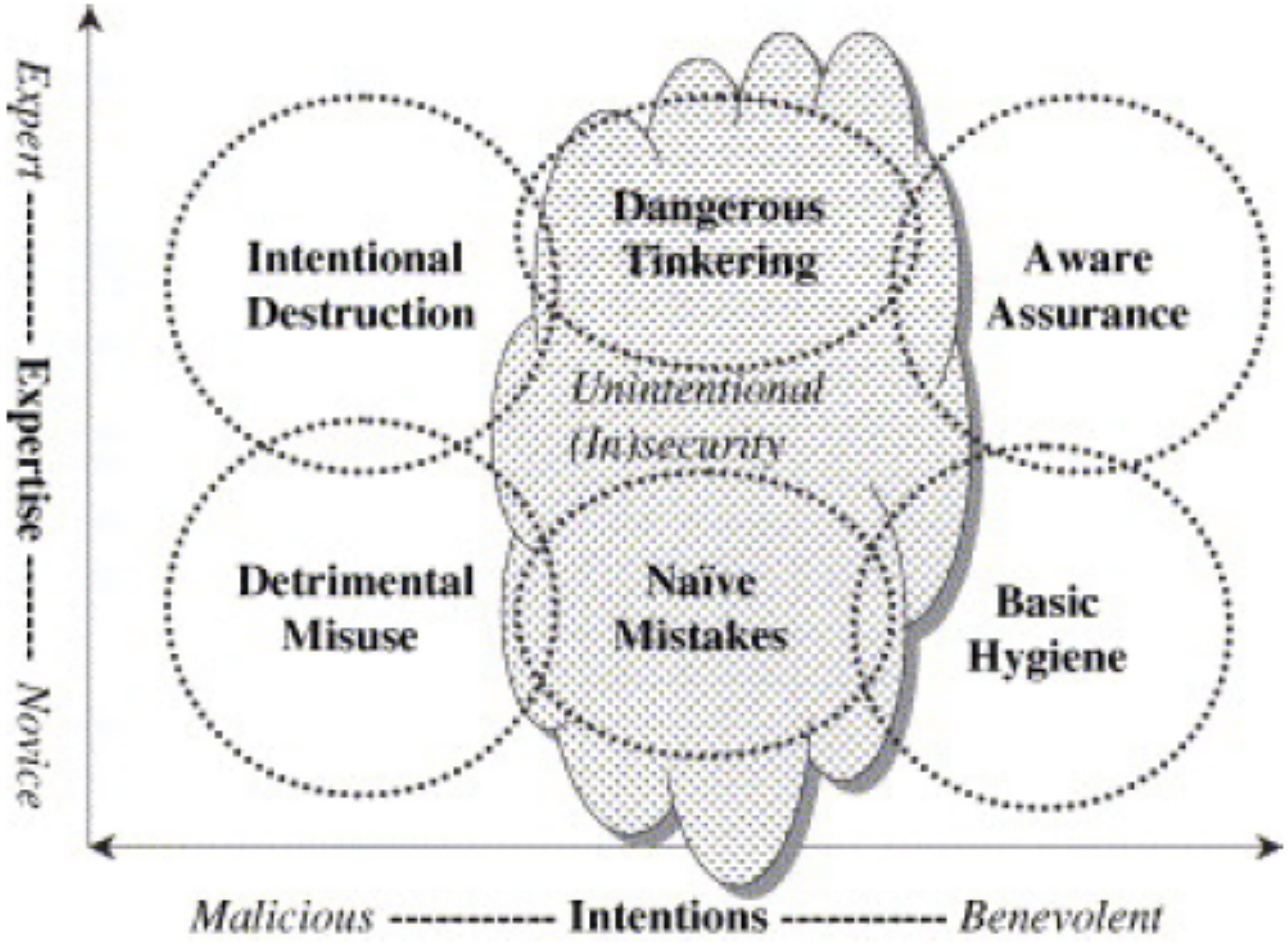
[Cranor, L. F., Garfinkel, S., *Security and usability: designing secure systems that people can use,* O'Reilly Media, 2005]

Human Factor Errors (con't)

S4Lab

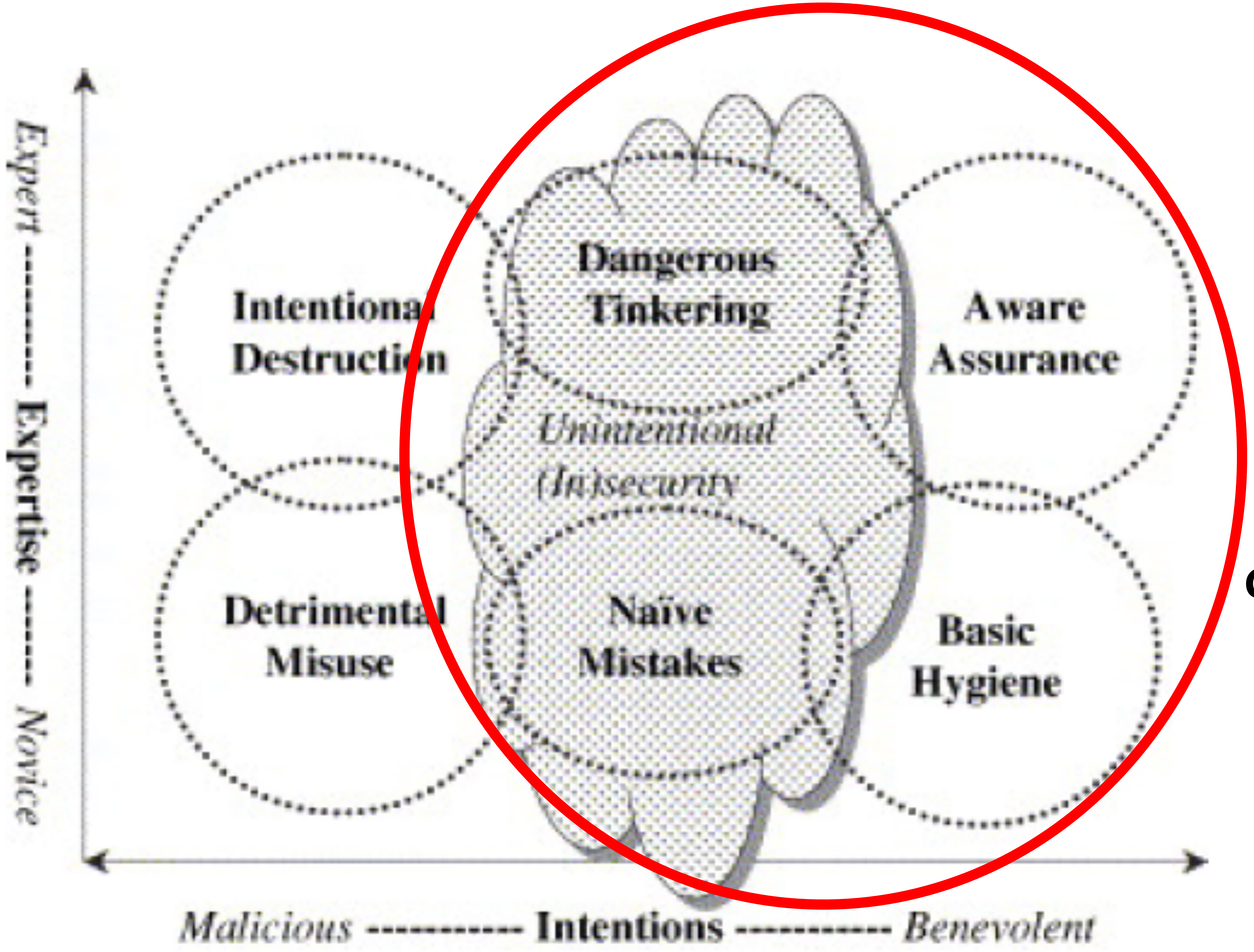
- Security behavior can also be described using a two-factor taxonomy:
  - Intentionality
  - Technical expertise

[Cranor, L. F., Garfinkel, S., *Security and usability: designing secure systems that people can use,* O'Reilly Media, 2005]



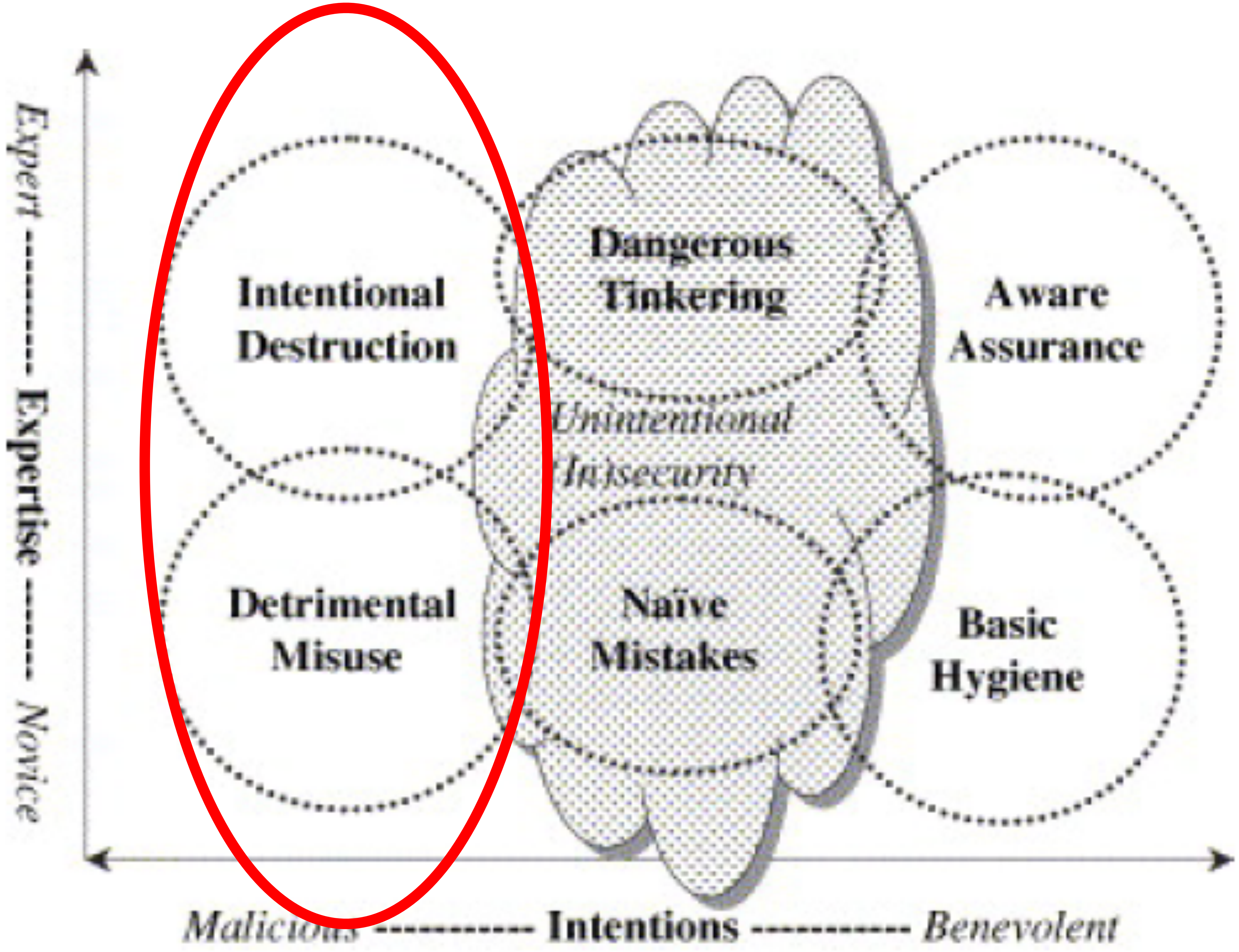
Parsons, K., et al., *Human factors and information security: individual, culture and security environment*. Def. Sci. and tech org, Australia, 2010]

S4Lab



Usability discussions discuss on these

Parsons, K., et al., *Human factors and information security: individual, culture and security environment*. Def. Sci. and tech org, Australia, 2010]



Usability discussions also affects these

Parsons, K., et al., *Human factors and information security: individual, culture and security environment*. Def. Sci. and tech org, Australia, 2010]

How to Balance Security and Usability

Security Is a Supporting Task

S4Lab

- Key to designing successful security applications.
  - Goals and tasks
- Human behavior —> goal driven
- Production tasks
  - Required to achieve the goal or produce the desired output.
- Supporting tasks
  - Enable production tasks to be carried out in the long run.
  - Or be carried out more efficiently, but not essential.
- Security tasks must be designed to support production tasks.
  - Should not conflict with production tasks (e.g. One’s performance)

[Cranor, L. F., Garfinkel, S., *Security and usability: designing secure systems that people can use,* O'Reilly Media, 2005]

Psychological Acceptability

S4Lab

- Security mechanism should not make accessing a resource more difficult, compared to when not present.
- In practice, a security mechanism should add as little as possible to the difficulty of the human performing some action.

[Image: Bauer, L., et al., Lessons learned from the deployment of a smartphone-based access-control system. SOUPS, 2007]

[Cranor, L. F., Garfinkel, S., *Security and usability: designing secure systems that people can use,* O'Reilly Media, 2005]

Exploit Differences Between Users and Bad Guys

S4Lab

- Dots when typing a password to protect from "shoulder surfing”.
- User would tend to pick an easier pass in order to avoid typos.
- Different perspectives:
  - The user is close to the screen
  - Eavesdropper is probably several feet away from the screen.
- Hence produce an interface that promotes complex passwords, while still leaving the eavesdropper in the dark.
  - Done in design of Tresor 2.2

[Cranor, L. F., Garfinkel, S., *Security and usability: designing secure systems that people can use,* O'Reilly Media, 2005]

Case study: Tresor2

S4Lab

- Tresor is a high security file encryption application.
- Makes it easy to type a long “passphrase" even if you make typos.
- As you type the password dots appear with a delay.
    - Revealing the last few characters for a few seconds as the user types, long enough for the user to catch a typo.
- Pressing Delete would delete the last character and reveal one more so that three would always be visible.
- Users can have longer passwords more easily

[Cranor, L. F., Garfinkel, S., *Security and usability: designing secure systems that people can use,* O'Reilly Media, 2005]

What else ?

S4Lab

- Exploit Differences in Physical Location
  - Our current "one size fits all" security systems tend to ignore that difference.
  - They arise from a single assumption: the bad guy may be standing behind you this minute!
- Vary Security with the Task.
- Increase Your Partnership with Users.
  - Trust the user.
  - Exploit the special skills of users.
  - Remove or reduce the user's burden.

[Cranor, L. F., Garfinkel, S., *Security and usability: designing secure systems that people can use,* O'Reilly Media, 2005]

Usability & Visibility

S4Lab

- Visibility is a powerful tool for aligning security and usability.
- Hidden properties, functionality, or data storage that is part of a complex system can make it more complex to use (less usable).
- So what to do?
    - Teaching users about hidden aspects of a system with significant effort.
    - An attractive alternative is to remove the opportunities for a system's visible state to be inconsistent with its internal state.

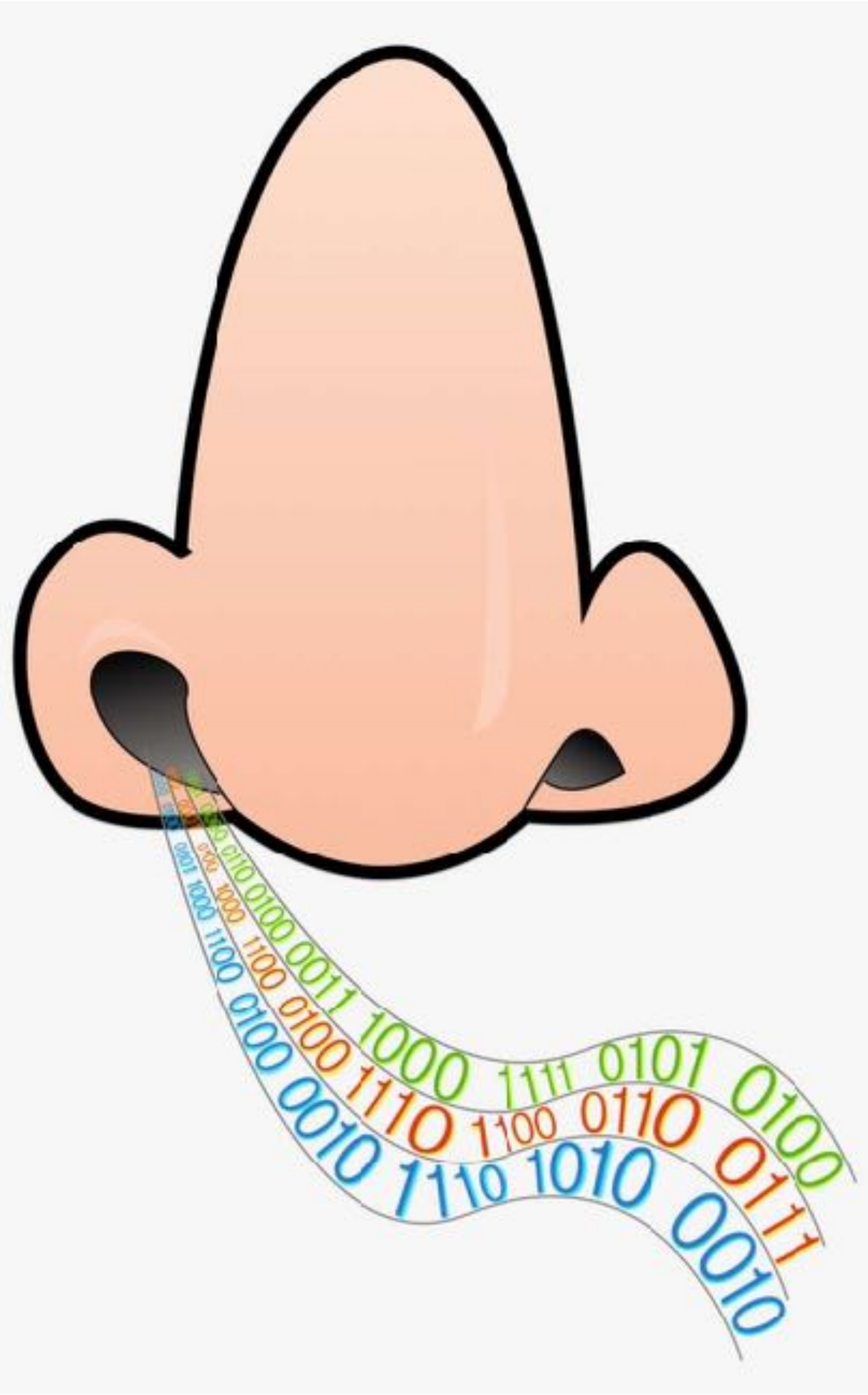
[Cranor, L. F., Garfinkel, S., *Security and usability: designing secure systems that people can use,* O'Reilly Media, 2005]

Usability smells: An analysis of developers' struggle with crypto libraries. Patnaik N., Hallett J., & Rashid A., SOUPS 2019.

Code smells

S4Lab

- Code smells are indicators that a piece of software code may be of lower quality than desired.
- The code may not be broken, but violating a design principle and may be fragile and prone to failure.

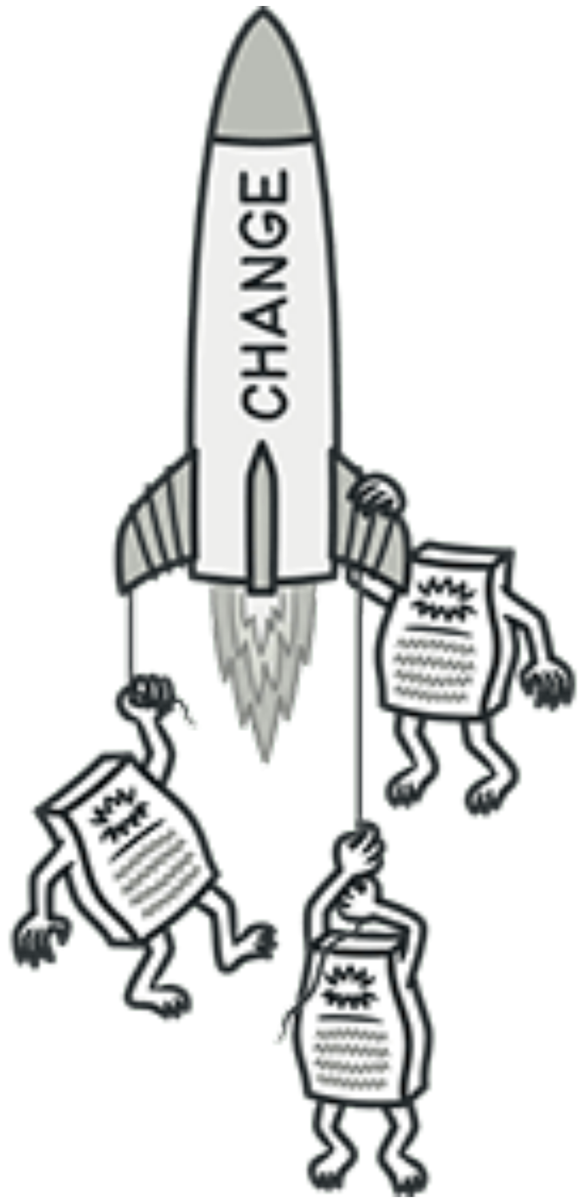


[Image: https://www.seekpng.com/]

[Patnaik, N., et al., Usability smells: An analysis of developers’ struggle with crypto libraries, SOUPS 2019.]

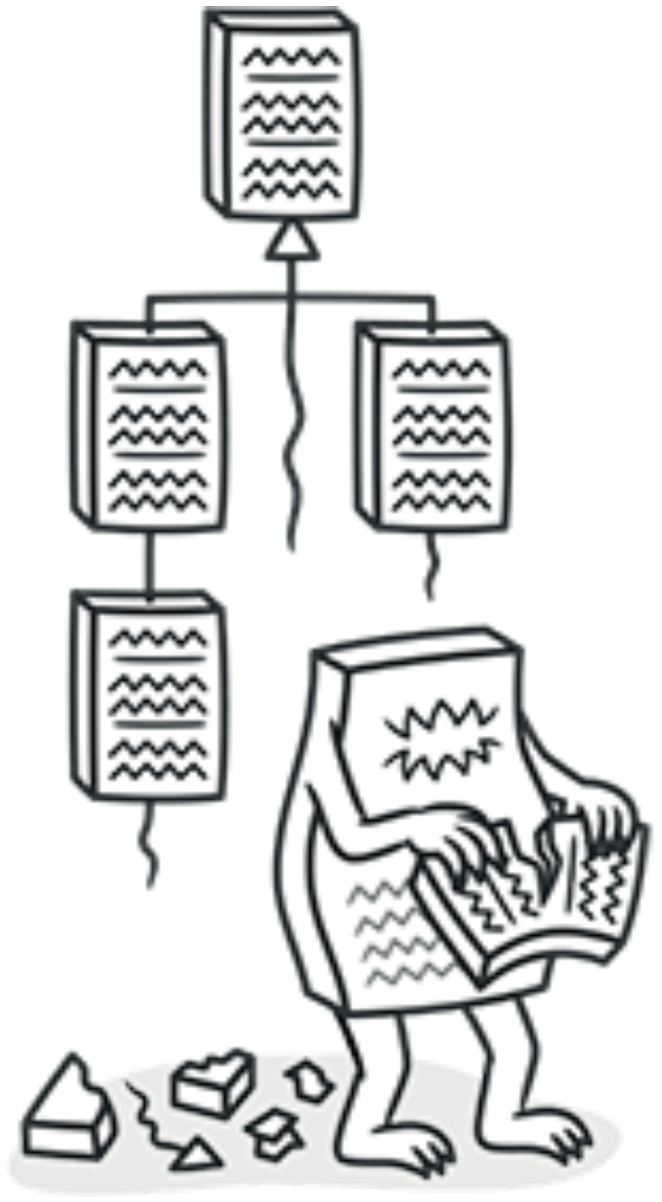
Examples

S4Lab



[Change Preventers]

These smells mean that if you need to change something in one place in your code, you have to make many changes in other places too. Program development becomes much more complicated and expensive as a result.



[Object-Orientation Abusers]

All these smells are incomplete or incorrect application of object-oriented programming principles.

[Couplers]

All the smells in this group contribute to excessive coupling between classes or show what happens if coupling is replaced by excessive delegation.

[https://refactoring.guru/refactoring/smells]

Usability smell

S4Lab

- A usability smell is an indicator that an interface may be difficult to use for its intended users.
- There have been multiple studies on usability smells in:
  - Graphical user interfaces
  - Library APIs (why?)
- The idea: the more usable API, The fewer questions about the basic usage
- Looking at a developer Q&A site, such as Stack Overflow
- 2,491 Stack Overflow questions to study about seven cryptographic libraries have been analyzed.

[Patnaik, N., et al., Usability smells: An analysis of developers' struggle with crypto libraries, SOUPS 2019.]

Thematic analysis

S4Lab

- Identifying 16 thematic issues and measure their prevalence across the different libraries.
- Relating these issues back to green and smith's usability principles
- Identifying four usability smells

Abstract Integrate cryptographic functionality into standard APIs so regular developers do not have to interact with cryptographic APIs in the first place.

Powerful Sufficiently powerful to satisfy both security and non-security requirements.

Comprehensible Easy to learn, even without cryptographic expertise.

Ergonomic Don't break the developer's paradigm.

Intuitive Easy to use, even without documentation.

Failing Hard to misuse. Incorrect use should lead to visible errors.

Safe Defaults should be safe and never ambiguous.

Testable Testing mode. If developers need to run tests they can reduce the security for convenience.

Readable Easy to read and maintain code that uses it/Updatability.

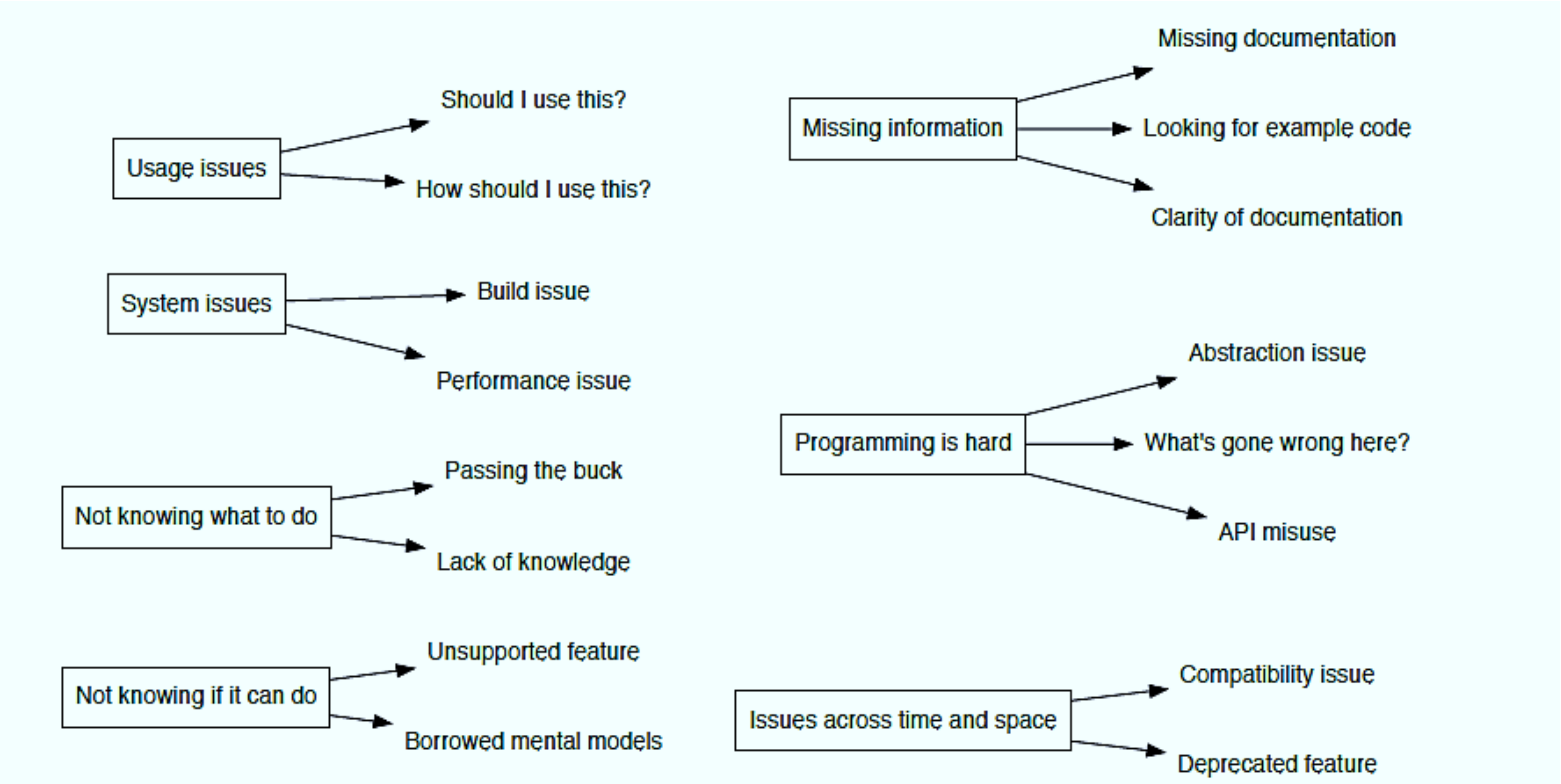
Explained Assist with/handle end-user interaction, and provide error messages where possible.

[Image: M. Green and M. Smith. Developers are not the enemy!: The need for usable security APIs. IEEE S&P, 2016]

[Patnaik, N., et al., Usability smells: An analysis of developers' struggle with crypto libraries, SOUPS 2019.]

The 16 issues identified through a thematic Analysis of Stack Overflow Questions

S4Lab



[Patnaik, N., et al., Usability smells: An analysis of developers' struggle with crypto libraries, SOUPS 2019.]

Final usability smells

S4Lab

Whiff	Issue	Abstract	Powerful	Comprehensible	Ergonomic	Intuitive	Failing	Safe	Testable	Readable	Explained
Need a super-sleuth	Missing Documentation				●						
	Example code				●		●				
	Clarity of documentation				●		●				
Confusion reigns	Should I use this?			●		●					
	How should I use this?	●	●			●					
	Abstraction issues	●				●					
	Borrowed mental models				●						
Needs a post-mortem	What's gone wrong here?				●	●				●	●
	Unsupported feature					●					
	API misuse						●	●			
	Deprecated feature						●				
Doesn't play well with others	Build issues								●		
	Compatibility issues		●								
	Performance issues										

[Patnaik, N., et al., Usability smells: An analysis of developers' struggle with crypto libraries, SOUPS 2019.]

Why Can't Johnny Fix Vulnerabilities: A Usability Evaluation of Static Analysis Tools for Security, Smith J., Do L. N. Q., and Murphy-Hill E., SOUPS 2020.

Why Can't Johnny Fix Vulnerabilities

S4Lab

- Static analysis tools enable developers to detect issues early in the development process.
- How usable are they?
- Static analysis tools can help prevent security incidents.
    - They must enable developers to resolve the defects they detect.
- Unfortunately, developers often struggle to interact with the interfaces of these tools.
- Leading to the proliferation of preventable vulnerabilities.

[Smith J., et al., Why Can't Johnny Fix Vulnerabilities: A Usability Evaluation of Static Analysis Tools for Security, SOUPS 2020]

Prerequisite: Some methods to evaluate the usability

- Two popular methods to evaluate the usability of user interfaces:
  - Heuristic evaluations
    - Most informal form.
    - Some experts judge about usability.
    - There are some principles (e.g. visibility, user control/freedom) to judge.
  - Cognitive walkthroughs
    - Cost-effective testing.
    - Is a user able to do a task easy? (task-driven)
    - There are some questions as guideline.
  - There are also other methods:
    - Consistency inspection, Pluralistic walkthrough , etc.

[Nielsen, J., Usability inspection methods. In Conference companion on Human factors in computing systems, 1994]

Methodology

- Two-phase method that combines the strengths of two usability evaluation techniques:
  - Cognitive walkthrough: evaluators simulate the tasks that real users would perform with a system.
  - Heuristic evaluation: evaluators systematically examine a system following a set of heuristics (as opposed to the task-driven approach in a cognitive walkthrough).
- Two evaluators
- User-study

[Smith J., et al., Why Can't Johnny Fix Vulnerabilities: A Usability Evaluation of Static Analysis Tools for Security, SOUPS 2020]

Phase 1: Task-Oriented Evaluation

S4Lab

- Particular task in mind: fixing as many errors as possible in a limited time.
- Following guidelines have been used:
  - Choose a vulnerability to inspect first.
  - Determine whether it is a true positive or a false positive.
  - Propose a fix to the vulnerability.
  - Assess the quality of the fix.
- To help us think critically about each tool, we used Sears' list of guiding questions (look in the paper)

[Smith J., et al., Why Can't Johnny Fix Vulnerabilities: A Usability Evaluation of Static Analysis Tools for Security, SOUPS 2020]

Phase 2: Free-Form Evaluation

S4Lab

- Where evaluators freely explore an entire system using a set of usability heuristics to identify issues.

Heuristic	Description
Preventing & Understanding Potential Attacks	Information about how an attack would exploit this vulnerability or what types of attacks are possible in this scenario.
Understanding Alternative Fixes & Approaches	Information about alternative ways to achieve the same functionality securely.
Assessing the Application of the Fix	Once a fix has been selected and/or applied, information about the application of that fix or assessing the quality of the fix.
Relationship Between Vulnerabilities	Information about how co-occurring vulnerabilities relate to each other.
Locating Information	Information that satisfies "where" questions. Searching for information in the code.
Control Flow & Call Information	Information about the callers and callees of potentially vulnerable methods.
Data Storage & Flow	Information about data collection, storage, its origins, and its destinations.
Code Background & Functionality	Information about the history and the functionality of the potentially vulnerable code.
Application Context / Usage	Information about how a piece of potentially vulnerable code fits into the larger application context (e.g., test code).
End-User Interaction	Information about sanitization/validation and input coming from users. Does the tool help show where input to the application is coming from?
Developer Planning & Self-Reflection	Information about the tool user reflecting on or organizing their work.
Understanding Concepts	Information about unfamiliar concepts that appear in the code or in the tool.
Confirming Expectations	Does the tool behave as expected?
Resources & Documentation	Additional information about help resources and documentation.
Understanding & Interacting with Tools	Information about accessing and making sense of tools available. Including, but not limited to the defect detection tool.
Vulnerability Severity & Rank	Information about the potential impact of vulnerabilities, including which vulnerabilities are potentially most impactful.
Notification Text	Textual information that an analysis tool provides and how that text relates to the potentially vulnerable code.

[Smith J., et al., Why Can't Johnny Fix Vulnerabilities: A Usability Evaluation of Static Analysis Tools for Security, SOUPS 2020]

Results

S4Lab

Theme	Subtheme
4.1 Missing Affordances	Managing Vulnerabilities Applying Fixes
4.2 Missing or Buried Information	Vulnerability Prioritization Fix Information
4.3 Scalability of Interface	Vulnerability Sorting Overlapping Vulnerabilities Scalable Visualizations
4.4 Inaccuracy of Analysis	
4.5 Code Disconnect	Mismatched Examples Immutable Code
4.6 Workflow Continuity	Tracking Progress Batch Processing

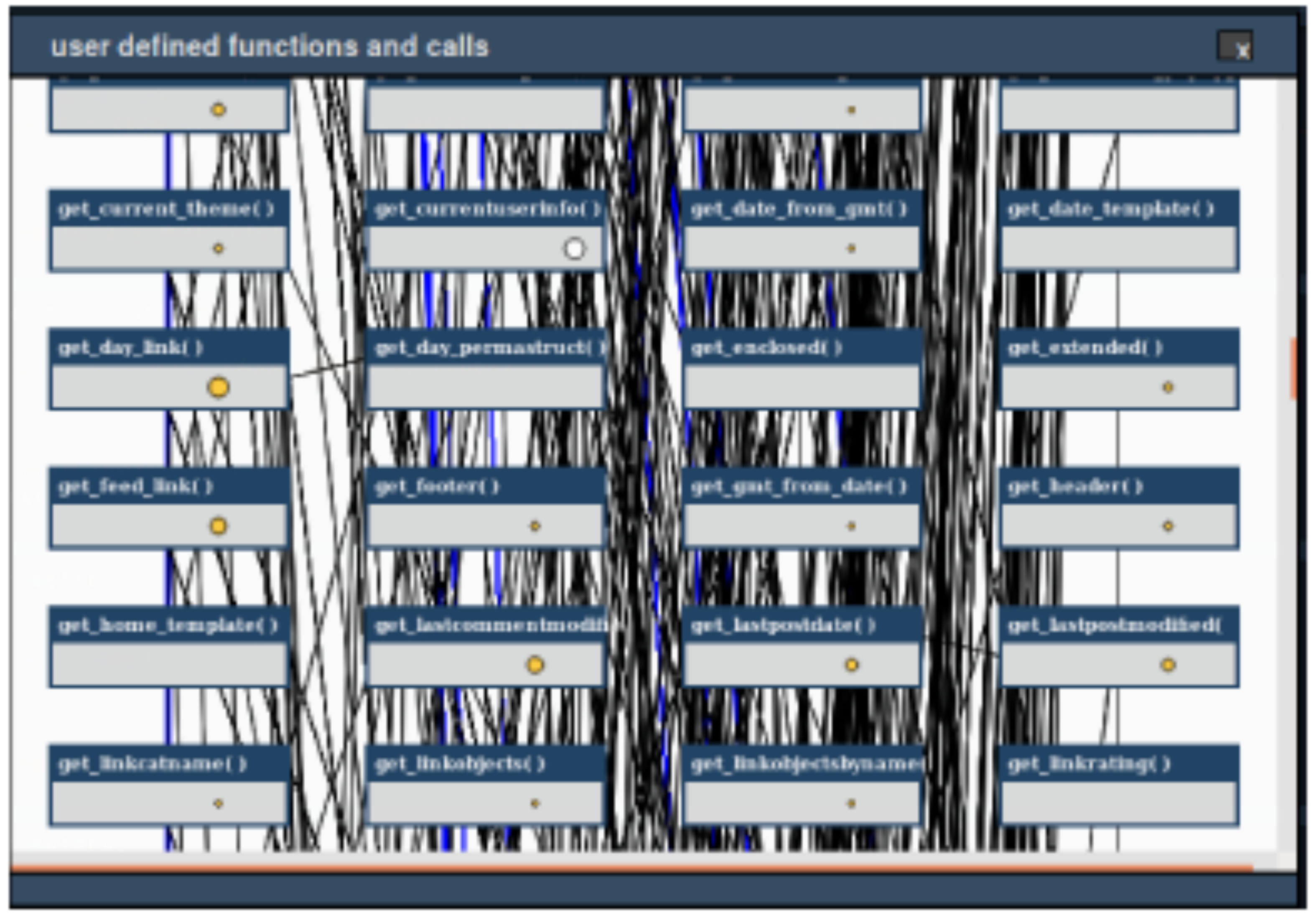


Figure 5: Scalability of RIPS' function view.

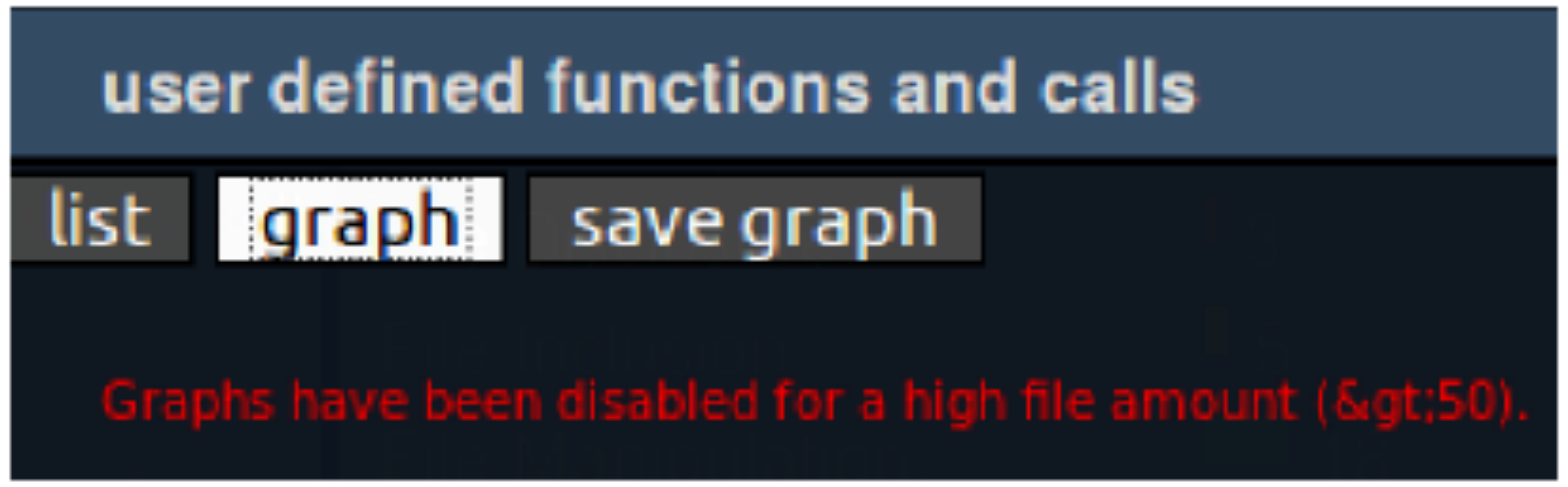


Figure 6: RIPS call graph visualization for more than 50 files.

[Smith J., et al., Why Can't Johnny Fix Vulnerabilities: A Usability Evaluation of Static Analysis Tools for Security, SOUPS 2020]

Further reading

S4Lab

- https://interactions.acm.org/archive/view/september-october-2007/the-problem-with-usability-problems1