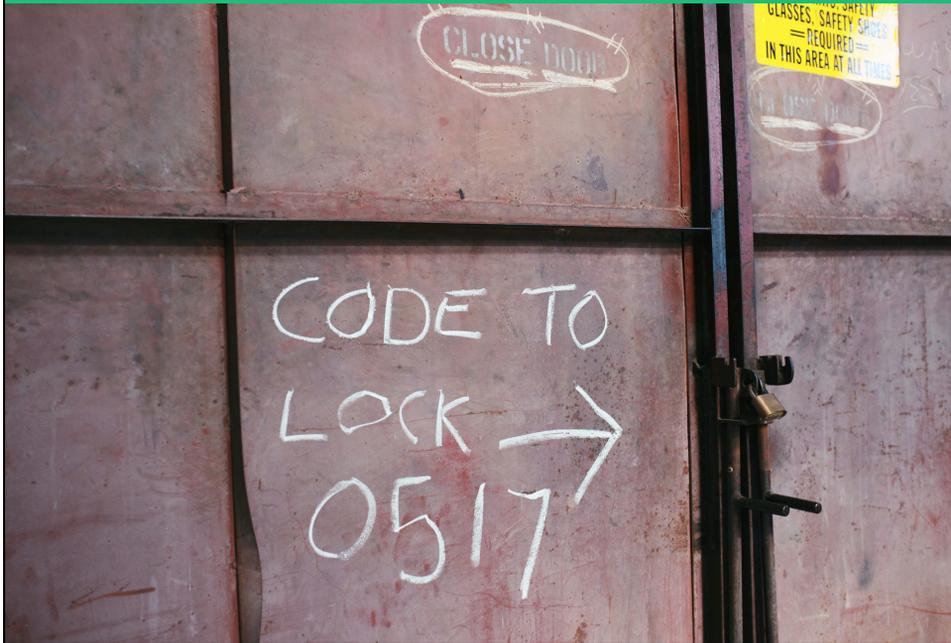


Users Are People Too

How to make your tools not suck for humans

Why human factors matter in security



[Gus]

Scout and I want to make the case that when you decide to develop a new piece of open-source software, particularly an application for security, you ought to begin by understanding the needs of the people who will use the software.

Those of us in the hacker and infosec communities are not completely in the dark about the impact of human factors. We take into account social engineering, shoulder-surfing, passwords on post-its stuck to the monitor.

But only thinking of human factors as attack vectors won't ensure your software gets used properly — or at all. You have to consider whether the security tools you are developing match the social systems they're operating in -- in this case, there are probably a number of people who need to access the door, and they may do so infrequently.

Ignoring the social systems or other human factors leads to compromises like these. Users will take whatever steps necessary to get done what they need to.



[Gus]

Listening to users' needs is going to mean putting aside biases you may have for wanting people to act by your standards for "logical behavior."

Because what's at stake may be, say, the next whistleblower's ability to speak out. Or the safety of a journalist in Vietnam. Or someone looking to leave an abusive relationship.

You may remember that Glenn Greenwald initially ignored Edward Snowden's requests that he talk to Snowden using encrypted email, because he figured it was too hard. In hindsight, this might look like a silly decision. It could have lost him one of the biggest stories of the past few years.

But Greenwald, like all of us, was working in a complex system of daily demands. He was working under the cultural pressures of journalism, with tight deadlines. He'd probably been approached plenty of times by nutjobs claiming to have a huge story; he had to have a threshold for weeding them out. His personal cost-benefit analysis was initially that it wasn't worth taking the time to learn a tool which, frankly, is really freakin' hard to use. Other sources were

not exactly clamoring to exchange public keys with him.

What usability and design teach us is the importance of respecting the complex human systems in which people are using software. User research is not just laboratory science: if you try to weed out weird-looking human behavior as “noise” or “outliers,” you will likely actually lose the data you’re trying to understand.

- "Greenwald and Miranda at Congress (cropped to Greenwald)" by Elza Fiúza / Agência Brasil - <http://agenciabrasil.ebc.com.br/galeria/2013-10-09/cpi-da-espionagem-ouve-jornalista-glenn-greenwald>. Licensed under CC BY 3.0 br via Wikimedia Commons - [https://commons.wikimedia.org/wiki/File:Greenwald_and_Miranda_at_Congress_\(cropped_to_Greenwald\).jpg#/media/File:Greenwald_and_Miranda_at_Congress_\(cropped_to_Greenwald\).jpg](https://commons.wikimedia.org/wiki/File:Greenwald_and_Miranda_at_Congress_(cropped_to_Greenwald).jpg#/media/File:Greenwald_and_Miranda_at_Congress_(cropped_to_Greenwald).jpg)

Hearing from your open source community ≠ getting feedback from users

Subscribing to the Enigmail Mailing List

Subscribe to enigmail-users by filling out the following form. You will be sent email requesting confirmation, to prevent others from gr hidden list, which means that the list of members is available only to the list administrator.

Your email address:	<input type="text"/>
Your name (optional):	<input type="text"/>
<small>You may enter a privacy password below. This provides only mild security, but should prevent others from messing with your subscription. Do not use a valuable password as it will occasionally be emailed back to you in cleartext.</small>	
<small>If you choose not to enter a password, one will be automatically generated for you, and it will be sent to you once you've confirmed your subscription. You can always request a mail-back of your password when you edit your personal options.</small>	
Pick a password:	<input type="text"/>
Reenter password to confirm:	<input type="text"/>
Which language do you prefer to display your messages?	English (USA)
Would you like to receive list mail batched in a daily digest?	<input checked="" type="radio"/> No <input type="radio"/> Yes
<input type="button" value="Subscribe"/>	

Screenshot of admin.hostpoint.ch/mailman/listinfo/enigmail-users_enigmail.net, powered by Mailman

[Gus]

At Simply Secure, we work primarily with open source privacy and security tools. There are a few big problems with the ways many open source secure tools projects get user feedback. One is that they tend to shoot first and ask questions later — they solicit feedback after developing tools, rather than before they start development.

The second has to do with their main channels for feedback. Mailing lists (like Enigmail is asking users to sign up for here), IRC, and GitHub are still their primary means of hearing about people's needs and issues.

Mailing lists require a huge personal investment on the part of the user -- receiving a lot of jargony mail and learning your culture of how/whether to ask for help (or RTFM) GitHub and IRC require many users to learn new tools on top of developing patience for jargon and cries of RTFM

These high barriers to entry make it so you repeatedly hear from That One Guy who has done a custom install of your software for Cyanogenmod on his Raspberry Pi and now has All Of The Needs

You already know you don't want to help that guy because he's causing his own problems. Do yourself a favor and find ways to listen to other people's feedback on

your software.

The alternative

Human-centered design puts users at the center of the design process.

- Develop **empathy**
- Design for **their needs**
- **Evaluate** your tool with them

You can do it too! Yes, even you, super-technical hacker!



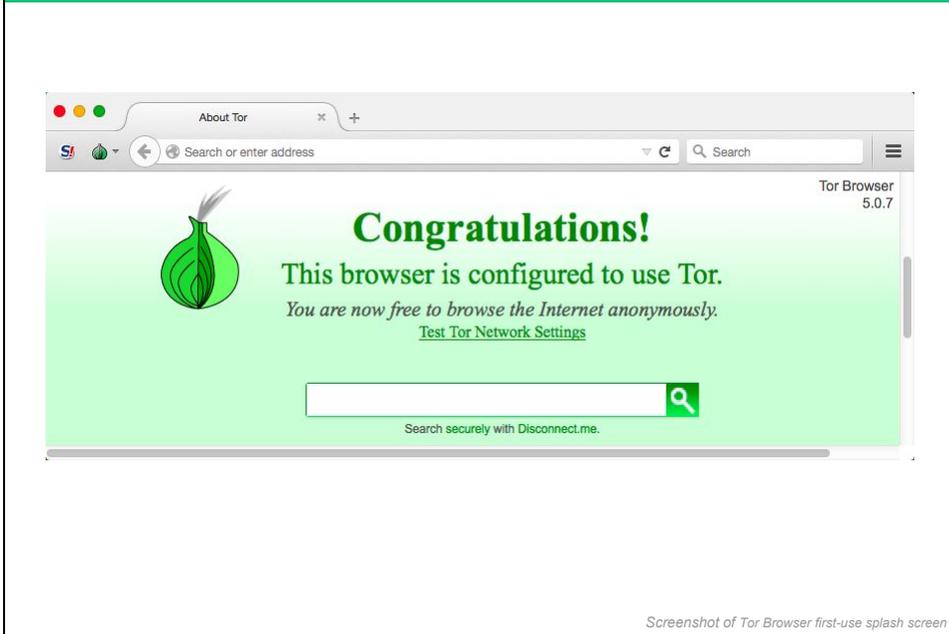
[Scout]

Usability & Design for Hackers: A few key terms

[Scout]

Putting our human-centered hat on, step one is admitting you're dealing with a different culture, and working to understand the language as the natives do.

User interface (UI)



[Scout]

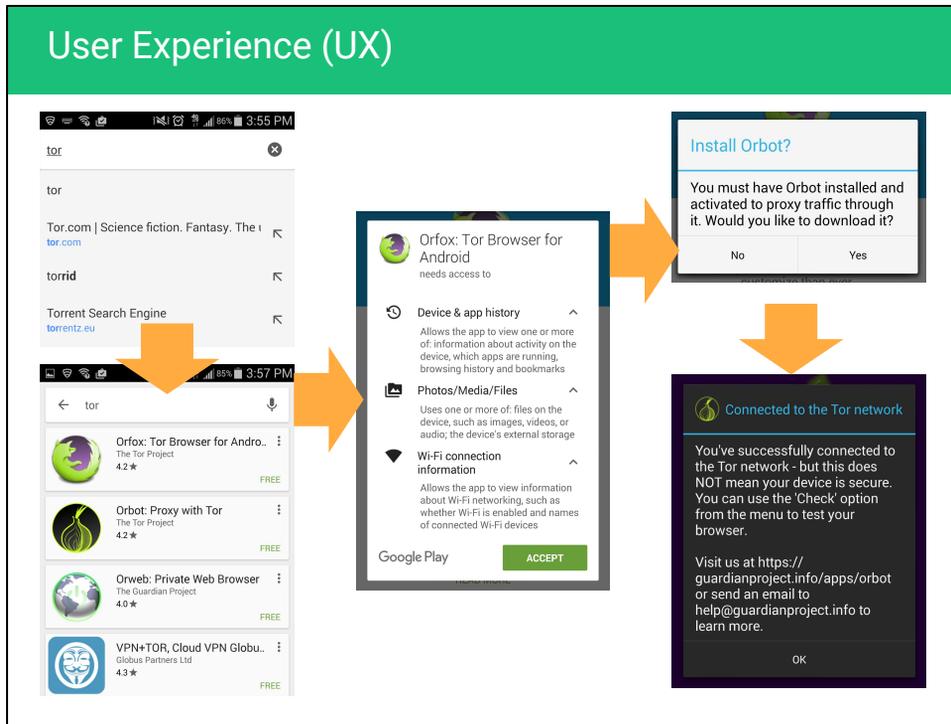
One term you're familiar with: User Interface. Here's an interface for the Tor Browser.

A user interface refers to the buttons, fields, text, images, and other elements which users see and can interact with.

We can critique an interface in a number of ways:

But an interface alone isn't the problem when we're talking about usability.

User Experience (UX)



[Gus]

So the sum total of the Tor user experience, and what users get out of it, is more than that interface. Consider Tor's new user flow: it might include searching in a browser, looking in the Google Play store and finding multiple things with the same name, making sense of a whole ton of permissions requests when you ask to install, realizing you have a piece missing, and getting a security warning you don't understand that directs you to another website. Other parts of user experience include lagging, troubleshooting, suspending, asking for help, sharing, customizing, syncing, and uninstalling.

The full experience is what user researchers study. And there's a reason one of Tor's project managers is devoting a lot of her time to understanding Tor's new user experience: they've identified they are losing big chunks of potential users at each of these steps.

Usability



[Scout]

This is a high-quality european car. It's functional. It's usable. It gets the job done. But it doesn't bring you joy when you use it.

Design



[Scout]

This is another high-quality European car. It doesn't just get the job done; it goes above and beyond, and brings a smile to your face while you're using it.

Usability is about basic functionality. It's necessary but not sufficient. Something that is truly well-designed goes further: it's elegant, and brings joy to its users.

UX research methods & tools

There exists a large, well-established, well-researched toolkit for crafting high-quality user experiences.

- A/B testing (but, privacy concerns)
- Semi-structured interviews
- Card sorts
- Ethnography
- Heuristic / expert evaluation
- Cognitive walkthrough
- Surveys
- Prototyping
- Task analysis
- Personas
- Co-design
- Paper prototypes
- Failure analysis
- Focus groups
- Eye-tracking
- etc.

[Scout]

There's a lot of other terminology that we could go into, and a lot of other things we could tell you about the practice of making great user experiences.

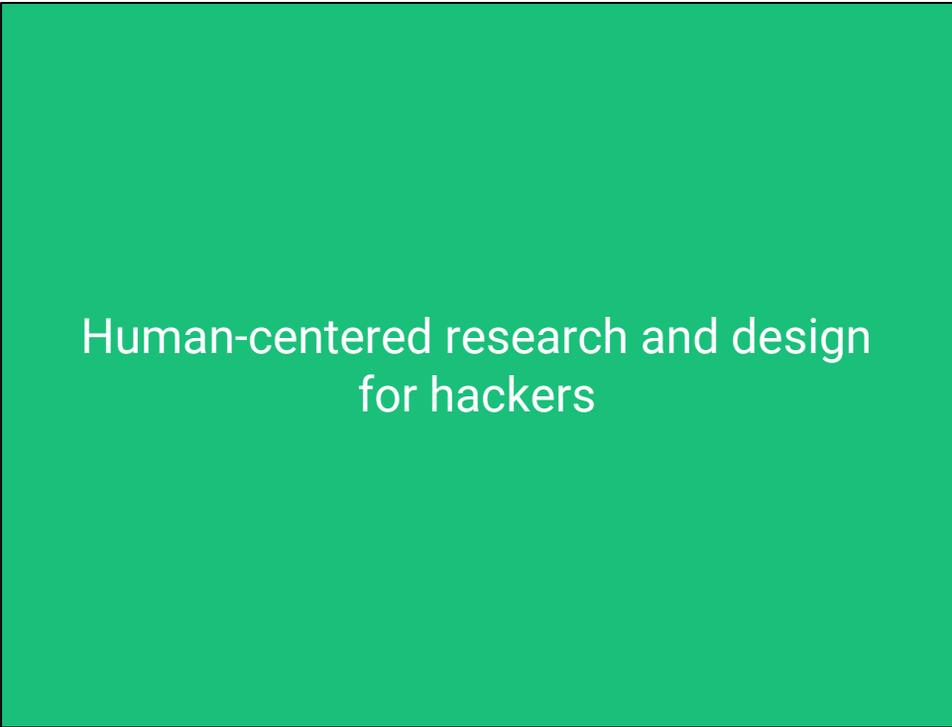
But, in the interest of time, I'll just say: user experience is a real field, people do real and serious work in it.

Let's talk about how we can adapt some of this into the work you do.

We don't want UX researchers designing crypto protocols.

Why do we assume technologists can design good user experiences without help?

[Scout]

A large green rectangle with a thin black border, centered on the page. Inside the rectangle, the text "Human-centered research and design for hackers" is written in white, centered horizontally and vertically.

Human-centered research and design for hackers

[Scout]

Steps to success (in an ideal world)

- 1) Know your users and assess their needs
- 2) Design for your users
- 3) Iteratively evaluate & improve your proposed UX
- 4) Ship & iterate with continuing user feedback

[Scout]

Your challenges



[Scout]

Very brief outline of the challenges that software projects face in doing human-centered work...

- you already have a tool that you've built and you're not interested in scrapping it and starting again from scratch.
- you might not have easy access to users
- you're not a ux professional: not knowing what questions to ask
- protecting the privacy of users – can't just vacuum up usage data like the big tech companies do

Steps to success (scrappy fixit mode)

- 1) Agree on your target users
- 2) Do an expert review of your UX to identify (& fix) low-hanging fruit
- 3) Interview real users
- 4) Build a model of your users and their needs
- 5) Smooth the path for user feedback
- 6) Iterate until you get it right

[Scout]

1. Know your users and their needs

CC BY-SA 3.0 [Nikolai Alekseev](#) @ Wikimedia Commons; CC BY-SA 2.0 [Rome Yandolin](#) @ Wikimedia Commons; screenshot from the Amnesty International Panic Button app page on the Google Play Store

[Gus]

- So, the first thing you need to do is define the population you are trying to reach. And this needs to be a detailed definition, because
 - secure is only secure in reference to a threat model
 - usable is only useable in reference to a user in specific circumstances.

It's important to understand the difference between threat models, use cases, and software features.

Threat models are prioritized sets of threats that a particular user faces

Use cases are ways in which *particular* users would like to address the needs they have in the real world -- which may include, *but are not limited to*, threats

Features are sets of of functionality that, when combined, address use cases

For instance: say you've identified that you'd like your security tool to support gay activists in Russia, who may be at high risk of physical attack, not to mention at risk of losing their jobs or being arrested. (That's the threat model.)

One **use case** might be the ability to notify people when they are arrested, in a way

that the police won't notice. Note this is not a feature itself -- you might meet this need in a couple of different ways.

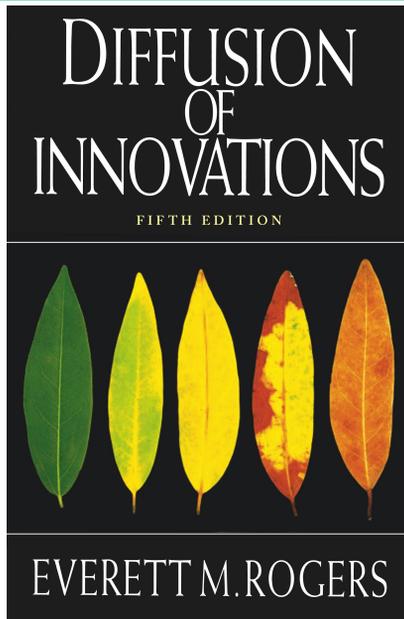
So, say you think you can meet this need with a "panic button" which will alert the contact's friends of their whereabouts. Given that the use case might be that they're targeted by a thug or police officer who not only might take their phone and use it to target their friends, but ALSO might physically attack them while doing so, the feature should be designed specifically for that use case — big old button, without a ton of "are you sure??!" dialogs or other things to get in the way when you need to hit it fast with one hand and then defend yourself, which also hides the app and what it's doing.

"Day of Kisses in Moscow" by Roma Yandolin - <http://www.flickr.com/photos/madw/9015242012/>. Licensed under CC BY-SA 2.0 via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Day_of_Kisses_in_Moscow.jpg#/media/File:Day_of_Kisses_in_Moscow.jpg

"MoscowPride2008-2" by Nikolai Alekseev at en.wikipedia. Licensed under CC BY-SA 3.0 via Wikimedia Commons - <https://commons.wikimedia.org/wiki/File:MoscowPride2008-2.jpg#/media/File:MoscowPride2008-2.jpg>

Panic Button screenshot from the Amnesty International Panic Button app page on the Google Play Store

Know the systems in which they work



- Who will adopt first?
- How will people observe others using your tool successfully?
- Is there a conflict between your tool and local cultural norms?
- Is it easy/enjoyable/satisfying/helpful compared to tools they already use?

[Gus]

Once you've identified who the users are, learn more about the world they live in

- Read Diffusion of Innovations. (or the Cliff Notes: Gladwell's The Tipping Point)
- Questions from Diffusion of Innovations:
 - Who do you expect to adopt first, and who do you expect to adopt later?
 - Not everyone is an early adopter. Others watch innovators.
 - How will people observe others using your tool successfully?
 - Is there a fundamental conflict between your tool and local cultural norms?
 - Sharing everything on social media is now a cultural norm, and that's at odds with the norms of the infosec community. That horse has left the barn, sorry! How will you design now?
 - How easy/enjoyable/satisfying/helpful is it compared to the tools they already use?
 - Each new setup screen adds to the likelihood they'll stop installing

2. Expert reviews



CC BY 4.0 [Erik Liljeröth](#)
@ Wikimedia Commons

[Gus]

Part of developing usability expertise is learning to think through the lens of other people's use cases, not just the uses you would like to put the tool to.

Expert reviews consists of someone skilled in UX thinking going through common expected workflows for the tool trying to put themselves in the shoes of the user. It makes sense to do this before you put the tool in front of any users, to be sure they're not just going to give up or be frustrated.

It's important to have someone who did not write the tool look at it. A designer, a researcher, a front-end person who has done user research.

One particularly principled form of expert review uses heuristics.

CC BY 4.0 [Erik Liljeröth](#) @ Wikimedia Commons, https://commons.wikimedia.org/wiki/File:NMA.0028271,_Fashion_Photo_by_Erik_Liljeröth_1954.jpg

Usability heuristics

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Copyright Jakob Nielsen @ Nielsen Norman Group

[Gus]

Usability heuristics are guidelines like the ones shown here.

You use these by walking through the workflow of a task that the users you've identified are likely to want to do, and check that each of these guidelines is met at each step.

So, for example: number nine

Error logs: useful for your use case, as a programmer. But consider the activist a few slides ago. Is that kind of error report useful for someone who's moving quickly? No. If there's no more visible notification than error logs, an application doesn't meet heuristic nine for the use case of an activist who doesn't have time to go hunting for the logs. (That might also fail #6, recognition versus recall: if the user doesn't often go looking for error logs and has to think hard to remember where they are, the software is putting too much load on the user's ability to remember.) What the activist from a few slides ago might need is some kind of well-worded alert, and an undo button.

So consider: What does your software currently do to help users recover from errors?

3. Interview real users



[Scout]

Define what “real users” means → not just fellow tech experts who speak your language and contribute to your Git repo. Your actual target user population.

Two potential goals, can be accomplished sequentially or in parallel:

- Get insights who they are and what their needs are
- Understand how the software works (or does not work) for them

LISTEN. Don't talk. Don't teach / educate – or at least not until you've spent a good time listening.

How to interview effectively

Don't ask leading questions

DON'T

- “How often do you encounter blocked websites?”

DO

- “Do you think you are blocked from getting to some websites? Why?”

[Gus]

- Questions to ask people when doing basic research. Examples include:
 - Talk to us about how you interact with technology in the course of your average day.
 - What are your concerns about who sees your information online?

How to interview effectively

Avoid jargon

DON'T

- “Do you encrypt?”

DO

- “How do you protect your privacy?”

[Gus]

- Weed jargon out of your questions
 - Jargon includes things like “HTTP.”

How to interview effectively

Use their terms

DON'T

- “Do you face censorship?”

DO

- not ask “how often are you censored” – to any Chinese person

[Gus]

- And “censorship!” Watch how fast some Chinese nationals run away from you when you casually bring up censorship.

How to interview effectively

Avoid stereotype threat and worry

DON'T

- Ask personal questions at the beginning (race, zip code, income, name)

DO

- ask them at the end if you need to

[Gus]

- Ask personal questions at the end, not the beginning

4. Build a model of your users and their needs



[Gus]

- After interviews: what next
- Making a model of your users helps to keep user needs in mind, in an abstract way that helps you make decisions, when you can't actually meet or go back to them in person
- Possible ways: personas, user stories, photos

omg this kitten

<https://www.flickr.com/photos/plasticrevolver/164351244>

User personas

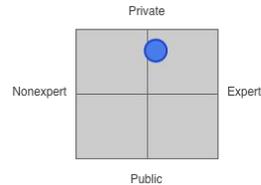
is.gd/securitypersonas



Fatima

“Help me communicate privately and safely with kindred spirits.”

- Mid 20s Egyptian trans woman
- Recent college grad
- Lives on her own
- Has come out to mother, who is in denial



Technology expertise level

- Mac super-user
- Vaguely aware of institutional surveillance, but not of entrapment

Technology use

- Desktop software
- Mobile apps
- Uses Facebook to manage multiple presentations of her gender (work colleagues)
- Suspends her account sometimes to keep people from posting to her account

Access locations

- Anywhere the mobile connection works
- Internet cafe

Threats from technology use

- Entrapment via engaging with a dating site
- Information getting to the “wrong” Facebook group
- Worries about ads tracking her and her friends

Physical threats

- Beating or death
- Arrest

Needs

- Love!
- Sex!
- Social interaction
- But also to keep her dating behavior separate on Facebook from other circles
- Keep her Mom happy
- And communicate in a safe, private environment

[Gus]

- After interviews: what next
- Making personas helps to keep user needs in mind when you can't actually meet or go back to them in person
- These ones are open source, free to use!

Personas: demographically similar



Shura

- Mid-20s
- Gay activist
- Mobile all the way; social media savvy
- Does not change privacy settings
- Goal: communicate to his community



Joseph

- 19 years old, college student
- Questioning: gay or trans?
- Mobile all the way; social media savvy
- Does not change privacy settings
- Goal: find like-minded community

[Gus]

- Here's some excerpts from the personas we develop.
- What personas are and are not
 - Not actual people
 - not stereotypes
 - task-based, not just demographic
- Take a look at these: here's what they'd look like on paper, with just simple quantifiable data.

Personas: qualitatively different



Shura

- Urban Russia
- Blogger/publisher under pseudonym
- Participant in online groups
- Risk he'll be jailed if outed by thugs researching his activity on social media
- Needs to remain anonymous BUT wants his pseudonym to be well-known



Joseph

- Suburban Idaho
- Fundamentalist Christian family
- At home, parents monitor all activity on shared computer
- Doesn't know terms like "transgender" or even "straight"
- Risk he'll be found out via search results, email, etc
- Likely to be kicked out of home and college if discovered

[Gus]

- Go over these
- Illustrative of key issues and needs
- Refer back to it as you decide which features to include: "is this what this person really needs?"

5. Smooth the path for user feedback



[Scout]

- What are the paths to user feedback that you support?
- Think of “providing feedback” as a use case of your project. Are your target users able to access it? What features do you have to support them?
- Pro tip: if your target users are not technologists, it’s not reasonable to ask them to learn how to use Github before providing feedback.
- A prominent email address on your website can be helpful as long as it’s monitored.
- A prominent feedback feature in your tool can be helpful as long as it’s monitored.

6. Iterate



[Gus]

- Once it's in the field, people will find new uses for it.
 - Amnesty Panic button has been re-used by midwestern families
 - Slide: kitten destroys dollhouse/Tale of Two Bad Mice
 - danah boyd's teens suspending accounts instead of logging out
 - and: long-necked giraffe->fluffy bunny



Lessons & War Stories

[Scout]

The perfect is the enemy of the good

Login to Your Account

Account Number:

Password: Use your mouse to enter your password on the keyboard below.
(Password is not case sensitive.)



Treasury Direct Login Screen image in the public domain @ [Wikipedia](#)

[Gus]

Balance user needs, including convenience, against attack likelihood.

Creating mass appeal



CC BY-NC-SA 3.0, morio @ commons.wikimedia.org

[Scout]

- Do I really want my tool to take off and improve the lives of all kinds of people?
 - yes. yes, of course I do!
 - this cannot be accomplished by sheer force of will

“But they say they want a purple pony”



CC BY-NC-SA 3.0, mamandil @ deviantart.com

[Scout]

“Let’s add a wizard”



CC BY-NC-SA 3.0, poitrus @ commons.wikimedia.org

[Scout]

“Let’s make this friction-free”



CC BY-NC-SA 3.0. androstachys @ commons.wikimedia.org

[Scout]

Conclusion

Conclusion

- Ignoring user needs will make your project vulnerable to all sorts of unexpected problems.
- UX research and design is a rich field full of useful approaches
- You can be scrappy in researching & designing for your users
 1. Agree on your target users
 2. Do an expert review of your UX to identify (& fix) low-hanging fruit
 3. Interview real users
 4. Build a model of your users and their needs
 5. Smooth the path for user feedback
 6. Iterate until you get it right

[Scout]

Resources to help you on your journey

- Simply Secure: we offer UX help to open-source projects with a security and/or privacy focus.
 - <https://simplysecure.org>
- Diffusion of Innovations (book mentioned earlier)
- Nielsen Norman Group articles
 - <https://www.nngroup.com>
- Second Muse needfinding framework
 - <http://internetfreedom.secondmuse.com/needfinding>
- Open-source resources for design
 - <https://github.com/opensourcedesign/resources>
 - <http://learndesignprinciples.com>

[Scout]