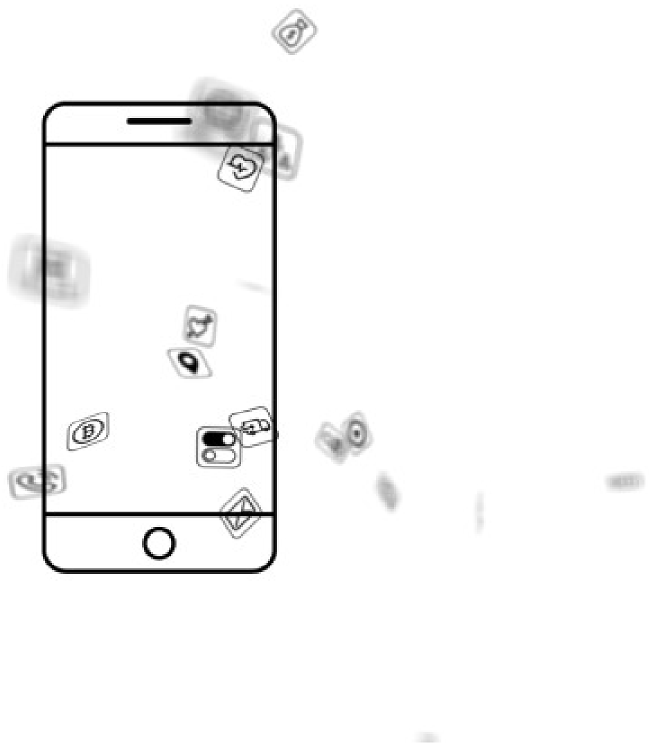
be trusted io

View on GitHub

Betrusted is a secure and private communications system. It gives users an evidence-based reason to believe that private matters are kept private.

Betrusted is more than just an app, and more than just a gadget – it is a co-designed hardware + software solution that provides safe defaults for everyday users. It’s also open source, empowering advanced users to analyze, extend and explore this secure mobile computer.

Developers can get started by visiting our documentation wiki.



1080p version

Our Core Principles

Betrusted is designed around the principles of transparency, simplicity, completeness, and self-sealing.

Transparency

Transparency is the bedrock of trust. Understanding what makes a thing tick gives us an evidence-based reason to trust that it works as intended. Betrusted is unique in that, instead of a black-box CPU chip, it uses reconfigurable hardware – an FPGA – for computation. This means you can compile our reference processor design from source, instead of simply having to accept on faith that this black epoxy rectangle contains precisely the circuits it advertises. Furthermore, we are at liberty to introduce countermeasures against known and future threats in the silicon supply chain by re-designing and re-compiling our processors.

Simplicity

Simplicity addresses the reality of only having 24 hours in a day. Even though the full source code for the Linux kernel and Firefox is published, nobody has the time to personally review every release for potential security problems; we simply trust that others have done a good job, because we have no other choice. Betrusted rolls the clock back to the early 2000's, when mobile computers were powerful enough to be useful for single tasks, while simple enough that individuals or small teams could build them from scratch. But don't worry, despite its simplicity, Betrusted's computational capability exceeds that of the Palm Pilot series. It's more on par with a Nintendo DS: sufficient for core security tasks such as authentication, instant messaging, and even end-to-end encrypted voice calls.

Completeness

Trust should begin at your fingertips and end at your eyes. Screen grabbers and keyboard loggers mean that chip-only trust solutions, such as TPMs, SGX, and secure enclaves are insufficient; we need a complete, end-to-end solution. Private keys are not the same as our private matters, and until we can encrypt data directly to and from our brains, the "analog hole" will be a real problem. With Betrusted, the complete loop of components from the keyboard to the display has been curated for transparency and simplicity, thus minimizing those attack surfaces that cannot be cryptographically secured.

Self-sealing

"Three can keep a secret, if two of them are dead". As Benjamin Franklin acutely observed, relying on trusted third parties to provision our keys is imprudent: self-sealing is the only way to keep a secret. Betrusted requires no special tools, NDAs, or third-party expertise to provision the system with secrets: it can generate and seal its own private keys on-chip in a transparent and open manner.

Our Solution

Security is hard to get right, and even the tiniest crack can be enough of a toehold for attackers to compromise the entire system.

Betrusted takes the approach of a co-designed hardware and software solution, because many of the cracks occur at the boundary of hardware and software. That being said, each element of Betrusted is potentially useful for non-security critical applications in their own right. Despite our integrated philosophy, each element is modularized so others can leverage our efforts for their own needs. Here is a brief overview of the elements that will make up Betrusted.

Hardware: A Precursor to Betrusted

The development hardware for Betrusted is available as an application-neutral platform called Precursor. It was developed in part with funding through the NGI0 PET Fund, a fund established by NLnet with financial support from the European Commission's Next Generation Internet programme.

At 7.2mm thin, Precursor is a pocketable mobile hardware development platform for secure applications. It embodies the principles of transparency, simplicity, completeness and self-sealing in a physical form. At its core lies not a CPU, but an FPGA, so that users are empowered to build their processors from scratch and know there are no flaws or backdoors in the architecture that could lead to security compromises.

To learn more about Precursor and how to support the project, please visit CrowdSupply, our distribution and crowdfunding partner, for more information.

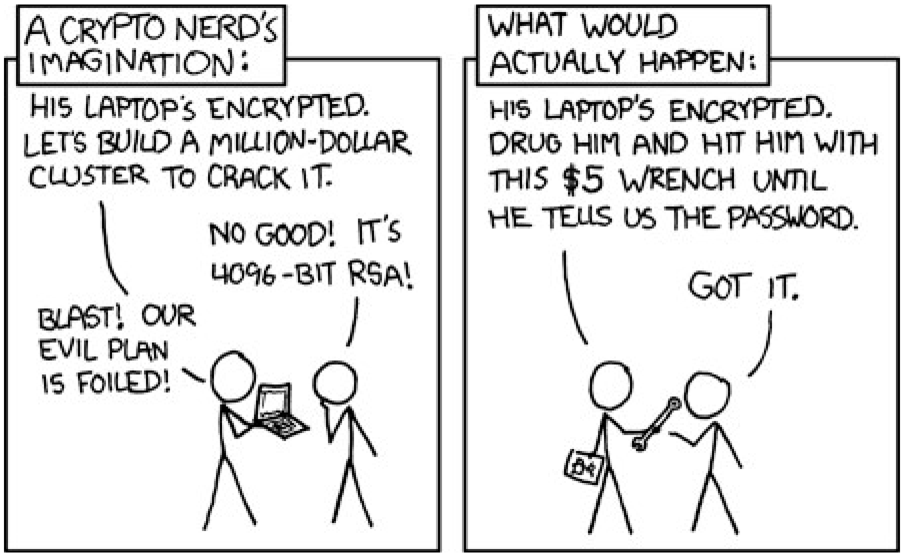
Xous: Extending Memory Safety to the OS

Xous is the microkernel OS for Betrusted. It's written in Rust, and not just because Rust is a trendy language. Xous extends Rust-style memory borrow checking to the operating system level. Unlike POSIX systems which treat everything like a file, Xous uses the *Message* as its fundamental data unit. Memory is shared between processes as a Message, with attributes of Borrow, Mutable Borrow, or Move. Processes "own" specific pages of memory, and attempts to access other memory results in a "panic". This is identical in concept to inter-thread communication in normal Rust. The hope is that by matching the OS level concepts around sharing memory between threads to the underlying language used to implement programs, we can eliminate or greatly reduce a large class of memory-safety related bugs. Of course, Xous can unsafely run binaries written in any language, but the OS-level bindings will be Rust-native, as opposed to C-native.

Xous' Message model for memory sharing also facilitates cross-platform development. This is because Messages can also be implemented by sending data over a network socket, meaning that the kernel and its services need not be restricted to run on a single physical machine. This enables easier debugging when actual hardware is unavailable. It also enables the use of traditional tools such as profilers on the kernel when running on the desktop. You can try it out now by following the directions at the Xous github page.

PDDB: Extending Your Safety to the "Filesystem"

The problem with building a device that is extremely good at keeping secrets is that it turns the user into the weakest link.


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

In practice, authorities need not even go so far as rubber-hose cryptanalysis to obtain passwords; simply demanding a user's passwords at Customs as a pre-condition for entry into a country has become a normalized deviance.

Therefore, enhanced security must come hand-in-hand with enhanced plausible deniability (PD) of any secrets that may or may not be contained within the system. PD is possible on traditional systems, but it is tricky; it is hard to train user to use the tools, and easy for apps to accidentally leave incriminating traces. It is also hard to say for sure on modern SSDs if data has been actually deleted or merely de-allocated.

The Plausibly Deniable DataBase (PDDB) is our answer-in-development to this problem. We plan to take advantage of the Message-oriented nature of Xous and our tight integration with the Precursor hardware layer to entirely do away with the notion of files and volumes. Instead, data is stored in a database as a set of key/value pairs that are associated with a security state. The set of user-visible data is transparently adjusted depending upon the set of PIN codes currently authorized by the user. We blend patterns of secure and insecure data access together all the way down to the hardware, while keeping end-user applications largely oblivious to the entire PD process. For example, the process of deleting a data set is identical to that of forgetting a PIN code. This makes it difficult to determine at the point of inspection if a complete set of PIN codes have been turned over.

Additional Planned Features

Making it easy for users to visually identify security-critical interactions is an essential countermeasure to phishing. We are in the process of building a simple graphical toolkit for building user interfaces that factor security into the rendering of graphics. It is particularly dangerous to allow applications to render full-screen bitmaps, as it gives compromised applications a way to trick users into divulging secrets. The graphics subsystem provided by Betrusted will put a special border around any bitmaps coming from an untrusted source, while reserving a region of the screen exclusively for icons that indicate the current security state of the system.

Development Status

Delivering a bespoke, hardware-to-user-app solution is a multi-year development process. As of September 2020, we're about two years into the project, and we are:

- Currently finalizing the Precursor hardware platform and preparing for crowdfunding
- In alpha stages of Xous development, with a PoC kernel running in emulation mode
- In the conceptual (napkin-sketch) stages of PDDB development
- In planning stages of higher-level features such as graphics and end user applications

Documentation

- The betrusted wiki aggregates all the technical documentation for the platform.
- The main betrusted github repository is the place to hunt for anything you didn't find at the link above.
- Video: 36C3 Introducing the core principles that motivate the hardware architecture
- Video: LCA20 Evolution and planning of the overall project, including software layers
- Video: FOSSNORTH20 Precursor overview talk
- Blog: Precursor SoC Tour
- Blog: Precursor Mainboard tour
- Blog: Other Precursor posts hardware security evaluation, braille keyboard, hackability, mechanical and PCB design aspects.
- Blog: Precursor + Renode

Tech morsels you may be interested in

- Avalanche Noise Source design notes
- Curve25519 Engine documentation

Who is behind betrusted?

The betrusted-io github repository's people page lists the developers that have elected to reveal their participation publicly.

The administrative contact for the betrusted.io project is Andrew 'bunnie' Huang (@bunniestudios/blog).

The Betrusted team is funded in part through the NGI0 PET Fund, a fund established by NLnet with financial support from the European Commission's Next Generation Internet programme, under the aegis of DG Communications Networks, Content and Technology under grant agreement No 825310.

Navigation