How not to design a privacy system: Reflections on the process behind the Freedom product

Adam Shostack, Ian Goldberg
{adam,ian}@zeroknowledge.com

1 Background

1.1 An Overview of the Freedom System

The Freedom System is designed to offer home users an easy way to use pseudonyms on the Internet. It is designed to resist a variety of attacks, up to, (but not including in version 1) national intelligence agencies. The system is designed to resist attacks by search warrant, by breaking into or subverting the system, and a variety of other attacks.

This design feature is implemented by having no computer outside the users control know about the source, destination, and content of any Internet packet sent. The system is technically similar, but not identical to, the NRL Onion Routing project.

The Freedom Network was also designed with "being able to get there from here" in mind. Although it would have been technically easier to implement if we had access to our own private network, over which we could control all traffic and protocols, the reality of the situation is that we would be deploying an overlay network over the existing Internet. This more easily allows additions to the (already worldwide) Freedom Network, and enables its incremental growth.

We also kept in mind user acceptability. Not only is it clearly in the interests of business to have as many users as possible on the Freedom Network, but having more users in a pseudonymous network adds to everyone's security, as well, since each user provides cover traffic for the others. To that end, we had a number of applicability goals. Users should not have to change their ISP in order to use pseudonyms online; we don't want to compete with the Internet; we want to enhance it. Users should be able to connect to our pseudonymous network via the Internet. Our system should support (in the sense that it should enable a user to use in a pseudonymous fashion) at least the most common Internet services, such as WWW, email, Usenet, and chat. Finally, the impact of the Freedom Network on the user's view of the Internet should be as small as possible. This means both that we should not impose an unacceptably large hit on bandwidth or latency, and also that we should try to keep the user's "view of the net" as close as possible to what it is without Freedom running. This last point involves a fundamental tradeoff, however, between the view of the net, and security. Users are notorious for using their computers and the Internet in an insecure manner, such as thoughtlessly running executables emailed to them, or found on the web. In designing the Freedom Network, we needed to trade off maintaining the user's security and privacy, versus (from his point of view) breaking functionality.

The first release of the Freedom system is an audacious and aggressive undertaking to provide privacy to consumers. It involved deploying an large scale infrastructure of routers and network management, privacy-enabled authorization and access control, and client software. We made many mistakes, some of which delayed the project, others caused us to ship less than we wanted to. Many of these were avoidable with careful planning, and the benefits of experience.

Better planning, design, and management would have allowed us to ship the system sooner, more closely matching the expectations we set, and allowing us to gain experience sooner.

We want to examine the things we did wrong and right, so as to improve our processes in the future.

2 What we did wrong

2.1 Lack of Development Process

This is a classic mistake, but one that people still make. We assumed that software development would be easy, and that we didn't need a process in place. Years of experience has shown this to be false, but we fell into the trap.

We strongly urge others building privacy systems to assume that it is hard work, and build a development process with experienced leaders who will drive schedules and quality.

2.2 Unclear and Changing Expectations

Ian knew what he wanted the protocols, etc. to look like, but that was only communicated to the developers via whiteboard sessions, not via formal requirements/specs documentation.

As a result, we have a bunch of places where the security isn't as good as we'd like for 1.0 (the DH agreement comes to mind).

Why? A couple of reasons. First, it likely had to do with the fact that Ian was physically in town only occasionally, and couldn't really legally contribute to crypto stuff while in the US. More importantly, he had never been involved in a formal spec process before, where he was not the one implementing his designs. Finally, we thought we didn't have time for it. Of course, the result was that it took a lot longer, anyway.

3 What we did right

3.1 Built the Business Model First

We didn't set out to build a privacy product without knowing how to sell it, and how to address (some of) the challenges that come with offering privacy in a privacy-unfriendly world. The founders of the company have developed, communicated and evolved a business model of psuedonymous activity that drives what we do.

3.2 Don't Rely on Trust

Bringing in Ian brought the company early credibility as a privacy company, but we didn't fall into the trap of relying on his good name. We designed systems so that Zero-Knowledge could be a malicious party, and you could still have privacy. This is desirable for lots of security reasons, and it enhances the position of the company.

3.3 Fix the Problems

This may seem obvious, but identifying and addressing issues of development process and leadership, rather than charging out to release, made things much better.

3.4 Resistance to Warrants

This may be a controversial feature.

Many people are not aware of the growing abuse of warrants in the court system. There are an increasing number of lawsuits filed and then dropped once a warrant has been issued. An example is the Raytheon case, where a number of users posted messages to a Yahoo! message board, critical of the company. Raytheon sued, got the identities of the 28 posters, and fired the 4 who worked for the company. The other 24 had their privacy violated as a side effect.

Similarly, the Digital Millennium Copyright Act allows a company to get a warrant based on an assertion that a pseudonym has violated their copyright. There is no due process protection for the privacy rights of the individual, and no penalty for being wrong. The Church of Scientology is aggressively using this feature to find the identity of its critics, who prefer to remain anonymous.

3.5 Audit Early, Audit Often

In any privacy system, specially one that involves cryptography, there are many places where bugs can creep in. Different styles of bugs need different ways of looking for them:

Lots of pairs of eyes

Many people should be auditing the code; both internal developers and external people with "fresh" eyes. These people are likely to find bugs such as memory leaks or buffer overflows.

At least one pair of crypto eyes

Most auditors are not qualified to find bugs in the cryptographic protocols or their implementations; you need at least one qualified crypto auditor to catch these kinds of things. (In our case, we spotted a (in retrospect amusing, but serious from a security standpoint) bug in the way the Diffie-Helman protocol messages were being sent.) The biggest catch is that, in general, broken cryptography still works; you can't find these problems at all by just testing - you have to look at the code.

Openness

Of course, the best way to have many people looking at your stuff is to make it open. At Zero-Knowledge, we've not only published our protocols and source, but we have also published a security analysis and a list of known attacks and other security issues. Doing so shows that we are being extremely open and honest about the capabilities, and limitations, of our product.