Official Technical Specifications

The Derpnet Team

April 10, 2011

1 Preamble

This is the documentation of standards for the Derpnet secure communication setup. The latest version of this document can be found in our git repository which is located at https://github.com/BUILDS-/Derpnet.

1.1 Licensing

1.2 Contributions and External Code

In accordance with our license, we will not depend on any proprietary code. While we prefer the GPL to other licenses, we recognize that this is not ideal for all developers and contributors. In order to not restrict contributions and existing code use, we will accept and use code under BSD, MIT, and other open source licenses, though for ease of distribution, this code may eventually be replaced with GPL-licensed code.

1.3 Declaration of Purpose

The Derpnet project is designed with three goals in mind: security, transparency, and usability. We believe that all three are necessary, and that any system missing even one of the three is divergent from ideal.

1.3.1 Security

For us, "security" encompasses both security in the traditional sense as well as anonymity. Traditionally, security has often come at the cost of anonymity. For example, email accounts have often been protected by "security questions" requiring personal information to answer. With this project, however, we are exploring the idea that this dependence on personal information from the "real world" which many would prefer to keep private may be unnecessary. Instead, one can create an identity completely separate from who they were born as, and attach everything to that, rather than to their name.

For example, consider a service providing secure email. This particular service performs all of its communication with clients over ssl/tls and the https protocol for web interfacing. To register an account, one need only entire the desired account name and password of said account, and it will be granted. The user can then use this account to register for other services based off of the email account thus created. Anonymity is protected, as there is no way to determine who the user is from the information provided to the email system (with the exception of IP-based analysis, which can be easily avoided if the user connected over Tor or similar service). Security in the traditional sense can also be supplied, as the email account that has been created can be used as the basis for authentication of other services. The user need not even trust the email provider with the text of their emails; with the ready availability of technologies such as PGP, this is no longer necessary.

The one down-side to this approach is that the email becomes a choke-point. That is, if the user forgets the passphrase to the email account, then, depending on the setup of the services that have been anchored upon it, the identity may be completely lost. Fortunately, many services today allow users to easily change the email basis for their identity, which does much to relieve this problem.

1.3.2 Transparency

For the threat vector that Derpnet is designed to protect against, security through obscurity is no security at all. An attacker is capable of reverse-engineering any component of a system, and many are skilled enough or have resources enough that the time to reverse a system is negligible. By this choice of attacker to defend against, we do not deny the

problem of lower-tech attacks such as keyboard logging for the purpose of passphrase sniffing, but rather suggest that defenses against such attacks as these be employed in addition to the protection a service like Derpnet provides.

Since there is thus nothing to be gained by the concealment of our system architecture and implementation, Derpnet is Open-Source Software under the terms of the GNU GPL. This means that any user is free to download the source code and tinker with it. The purpose of our core team is to function as both development leads and a semi-regulatory body so that our code remains cohesive enough to be useful while adding new features, bug-fixes, and the like.

1.3.3 Usability

We judge any system that cannot be used, no matter how secure or transparent, to be without purpose. It is our task as developers to ensure that our system is not only useful to but also usable by all of our users. To that end, code integrity is our responsibility, as well as adherence to standards. To express that in practical terms, if we claim code should compile on your setup, and it does not, it is our fault, not yours. In addition, we have provided multiple means of access to all of our subsystems, as will be discussed below.

2 Architecture

In accordance with our attempt toward transparency, this section attempts to explain the infrastructure that will power the initial Derpnet. We make use of the anonymity-model discussed in the security section.

2.1 Email Server

We intend to use an email service as the basis for all of the other services we provide. There will be multiple ways to access as an email account: a secure (https) webmail interface will exist, and additionally we will provide secure pop and / or imap access with smtp message delivery. Creating an account on this system will require no other information than the desired username and passkey, though other procedures to eliminate abuse (such as idle account deletion, or the dreaded CAPTCHA) may require implementation and use as well.

To encourage proper use of services, we encourage users to make use of the email thus created as the basis for all other services, though, in the interests of both usability and

security, we will not require it.

2.2 The IRC Protocol

Internet Relay Chat is one of the oldest forms of internet-based inter-user communication, and is unique for its room-based architecture. As it has been in use for many years, modifications to the basic protocol have developed, some useful, and some less so. Perhaps the most important has been the implementation of ssl connections (typically on port 6697). In addition, many servers provide NickServ implementations which allow users to authenticate themselves, which increases security without sacrificing any more anonymity than an email address. However, with the increased use of Tor and other proxies, many networks have felt the need to block the use of such services in order to prevent their abuse for DDoS and other attacks.

Obviously, if it were possible to use Tor and other anonymizing networks to connect to an IRC server without possibility of their abuse, it would be beneficial from a security perspective. We believe that we can use NickServ to solve this problem. By requiring all users to be registered with NickServ (depositing an email for confirmation purposes), there becomes no inherent difference between a Tor / proxied connection and any other connection. Sure, the user must identify, but we think, after our own experience with IRC, that most serious users will want to do this anyway, so the inconvenience is minimal.

In addition, we define three methods of connection to IRC: legacy, encrypted, and secure.

2.2.1 Legacy IRC Connection

This method will attempt to comply with the IRC RFC documents as much as is possible without compromise to the security of the servers themselves. We will include the NickServ implementation discussed above in this mode of connection. This "legacy" will operate on port 6667, which has become the *de facto* standard for unencrypted IRC. This will only be a temporary means of IRC connection, and, as it is considered insecure, will be depricated upon readiness of the "encrypted" connection schema and removed entirely upon readiness of the "secure" connection schema.

2.2.2 Encrypted IRC Connection

This mode of connection will operate off 6697, which has become standard, as well as potentially other ports. This mode of interface is designed to allow existing clients, such as irssi, pidgin/Adium/finch, and others that have given security at least a token amount

of work, to be able to use the Derpnet infrastructure. All connections will be encrypted ssl, and will attempt to provide as much of the "secure" connection schema as is possible without breaking compatability with these clients.

2.2.3 Secure IRC Connection

This is our most secure connection scheme. All connections to and from the server will be encrypted using OTR. In addition, to both increase security and reduce server load, where possible the two participants of a private query will be directly connected over OTR. We will again use the NickServ scheme mentioned above to prevent DDoS.

Finally, we will introduce a new mode for channels similar to +R (which indicates that only users registered users can join) such that only users that are connected over this secure method may join. (TODO: decide what mode this should be. We could commandeer +R since only registered users can exist on-network anyway, or make a new one to eliminate confusion.)

3 Unresolved Questions

In the order that they occurred to me in:

- What should the new mode be?
- What of the channel prefixes from the RFC do we want to implement?
- Are we C++ exclusive, or do we allow language mixing?

4 Further Reading

We have included the IRC RFCs in our repository. We also recommend perusal of any documents you can find on OTR, as well as spending some time on an IRC network to get a feel for how the system as a whole behaves. In addition, any expertise / insight into email systems would be greatly appreciated.

5 Credits

Derpnet is: $\alpha, \beta, \gamma, \delta, \epsilon$.