Installing D'Agents: Release 2.0

Robert S. Gray
Thayer School of Engineering
Hanover, NH 03755
E-mail: robert.s.gray@dartmouth.edu

March 11, 1998

1 Introduction

This document describes how to install D'Agents 2.0 *without* installing the Pretty Good Privacy (PGP) program. Since D'Agents uses PGP to authenticate incoming agents (and assign access restrictions based on this authentication), installing without PGP means that all agents will be treated as anonymous agents. Thus, installing D'Agents 2.0 without PGP will give you much better performance than with D'Agents 1.1, but only a small amount of additional security.

If you want full security, you should *stop reading now and switch to the other set of installation instructions*, which describe how to install PGP and D'Agents together.

If you need any help with installation, please send e-mail to robert.s.gray@dartmouth.edu.

2 Installation

The source code for D'Agents 2.0 is packaged as a tar file

`agent2.0.tar.gz`

which can be obtained via our World Wide Web page

`http://www.cs.dartmouth.edu/~agent`

or via anonymous FTP

`ftp:://ftp.cs.dartmouth.edu/pub/agents/`

The source code distribution supports Tcl agents only. Support for Python, Java and Scheme agents will be publicly released at a later date. In addition, the distribution contains *all* necessary code. You do not need the source code for the standard Tcl/Tk interpreters, and you do not need to install the standard Tcl/Tk interpreters on your system.

D'Agents 2.0 is a Unix-only system. It is known to compile and run correctly on the following platforms:

- Linux 2.0.30
- AIX 3.2.5
- Irix 6.2
- Digital OSF1 V4.0
- FreeBsd 2.1
- Solaris 2.5.1

It is known to compile and run correctly with the following compilers:

- GNU gcc/g++ (versions 2.6.3, 2.7.2 and 2.8.0)
- SGI CC (Edison Design Group C/C++ Front End version 2.19)
- Sun CC (version 4.2)

D'Agents 2.0 should be easily portable to any modern Unix, and since it does not use C++ exceptions or templates, most C++ compilers should be able to compile it. Porting to non-Unix operating systems, such as Windows NT or Windows 95, will require a significant amount of work.

To install D'Agents 2.0:

1. You have chosen to install D'Agents 2.0 without installing PGP. Thus, although D'Agents 2.0 will provide enough security for experimentation, research, and initial development, it will not fully protect a machine against malicious agents. Therefore, since incoming agents run under the same userid as the D'Agents server, you should *not* run the server as *root* or as any userid that has authority to access or overwrite sensitive data. In fact, it is highly recommended that you create a new account for the D'Agents system and run the server from this account. The account should have only as much authority as is needed for the desired prototype application(s). In this document, I will assume that you create an account with username *dagents*. For the greatest peace of mind, you should also create a new group, and make username *dagents* a member of that group only.

2. Log onto your new *dagents* account, download agent2.0.tar.gz, uncompress and untar. This process will create a directory called *agent2.0*. Read the README file that is in the *agent2.0* directory before you continue.

3. Edit the *Makefile* and specify the desired compilation options. Most of the options should be self-explanatory.

   SERVER_TCP_PORT specifies the TCP/IP port that the server uses to communicate with agents on remote machines. You *must* use the same TCP/IP port on every machine. SOCKET_NAME specifies the filename of the Unix domain socket that the server uses to communicate with agents on the local machine.

   USERNAME and GROUPNAME specify the user and group to which the server will switch if it is started as root. USERNAME should be the username of the account that you created in step 1 (e.g, *dagents*). GROUPNAME should be the associated group.

   LOCAL_MACHINE specifies the name of the machine on which you are installing the server. This information is not used during compilation, but it is used to generate some default configuration files.

   DATA_DIRECTORY specifies the directory in which the server will create the Unix domain socket SOCKET_NAME. In addition, the default configuration files will be installed into DATA_DIRECTORY.

   ADMIN_KEYNAME is the PGP keyname of the server administrator. Since you are installing without PGP, the value of ADMIN_KEYNAME can be anything.

   TEMP_DIRECTORY specifies the directory in which the server will create its temporary files.

   SOURCE_PREFIX is the source code directory (i.e., the directory that contains the Makefile that you are currently editing). BUILD_PREFIX is the build directory (i.e., the directory in which .o files will be stored during compilation). Both SOURCE_PREFIX and BUILD_PREFIX should be fully qualified.

   Pay special attention to TCL_LIBRARY and TK_LIBRARY, which specify the directories where the Tcl/Tk interpreters expect to find their initialization scripts. TCL_LIBRARY and TK_LIBRARY should resolve to the same directories as TCL_INSTALL and TK_INSTALL respectively.

   Finally, system management will be much easier if all of the installation subdirectories are in the same top-level directory. For example, in the default Makefile, all of the installation subdirectories are in /usr/agenttcl.

4. Compile the modified Tcl interpreter (*agent*) by typing

   ```
   make tcl_agent
   ```

   Compile the modified Tk interpreter (*agent-tk*) by typing

```
make tk_agent
```

Compile the server (*agentd*) by typing

```
make serv
```

Like many Makefiles, the Makefile does not detect and recover from as many errors as it should. Watch the output for anything that looks like an error message.

5. Customize the configuration files and Tcl scripts by typing

   ```
   make customize
   ```

   This step creates reasonable default configuration files and changes all the Tcl scripts so that the first line points to the correct executable, etc.

6. Install the system by typing

   ```
   make install
   ```

   This installs the executables, the libraries and the configuration files. If you want to install only the executables and libraries (i.e., if you do not want to overwrite existing configuration files), type

   ```
   make install-code
   ```

   instead.

7. Type

   ```
   make ranbits
   ```

   to generate a random bits file that the server will use to initialize its internal random number generator. Although this step takes a while, *it is worth it*, especially if you later decide to install PGP and turn on full security.

8. Add the directory that contains the *agent* and *agent-tk* interpreters to the PATH environment variable on the *dagents* account – i.e., add EXEC_INSTALL to the PATH variable. Add the directory that contains the shared libraries to the LD_LIBRARY_PATH environment variable on the D'Agents account – i.e., add LIB_INSTALL to the LD_LIBRARY_PATH variable. If you do not want other users to be able to run the interpreters, go to the EXEC_INSTALL directory and turn off the group and other permissions. If you do want other users to be able to run the interpreters, make sure

that the other users know about adding EXEC_INSTALL to their PATH and LIB_INSTALL to their LD_LIBRARY_PATH. Of couse, you can go halfway instead, and allow only a certain group of users to execute the interpreters (e.g., perhaps the special agent group that you created back in step 1).

The entire installation process must be repeated for every distinct architecture on which you want to run the D'Agents agent system. On duplicate architectures, you do not need to recompile, but you will need to do everything else (i.e., edit the Makefile, *make customize*, *make install*, *make ranbits*, etc.). On every machine, you should use a different set of installation directories, even if two or more of the machines are the same architecture. Although it is possible to share the *bin*, *lib* and *scripts* subdirectories, it gets confusing quickly.

3 Running the server

The server must run on every machine to which you want to send mobile agents. For each machine:

1. Log onto the *dagents* account. Remember that you want the server to run with the minimal permissions of the *dagents* account.

2. Edit the server configuration file *agentd.conf*. This configuration file is in the DATA_DIRECTORY that you specified in the Makefile.

   Since you are not using PGP, you are only concerned with a few sections of this file. First, you should change the *Encryption* field so that it says *no*. This field tells the server whether or not PGP is available. Second, you should add an *AllowedMachine* line for each machine from which the server should accept agents (and inter-agent communication). As in D'Agents 1.1, the only criteria for acceptance when you are not using PGP is the apparent IP address of the sending machine. Finally, at the end of the file, there is a set of five fields, *AnonStartWall*, *AnonStartCpu*, *AnonStartJumps*, *AnonStartChildren*, and *AnonStartDepth*, that specify initial resource limits. Since you are not using PGP, every agent is an anonymous field, and these initial resource limits can be set relatively high. If you get error messages saying that an agent's current permit does not allow it to jump, an agent has violated its global wall-time permit, etc., the resource limits are not high enough. You will either have to increase the resource limits in each server configuration file, or the problematic agent will have to use the `require` command to ask for a higher limit.

   The fields *Owner*, *Admin*, *AllowSigned*, *AllowedUser*, *TrustedMachine* and *ReservedName* probably look important. Since you are not using PGP, however, and all agents are anonymous, these fields have no effect. If you decide to use PGP later, these fields will become critical.[1]

3. Change into the server directory. The server directory is the SERV_INSTALL directory that you specified in the Makefile.

4. Start the server by typing

   ```
   ./start_agentd
   ```

   You will see output that looks like the following:

   ```
   Tue Mar 10 12:19:06 1998 - loadConfig: initial:
     INFO: AllowedMachine "localhost"
   Tue Mar 10 12:19:06 1998 - loadConfig: initial:
     INFO: AllowedMachine "grumpy.dartmouth.edu"
   Tue Mar 10 12:19:06 1998 - loadConfig: initial:
     INFO: TrustedMachine "localhost"
   Tue Mar 10 12:19:06 1998 - loadConfig: initial:
     INFO: TrustedMachine "grumpy.dartmouth.edu"
   Tue Mar 10 12:19:06 1998 - loadConfig: initial:
     INFO: name "resource-manager" reserved for owner
     "Agent Tcl administrator <cs.dartmouth.edu>"
   Tue Mar 10 12:19:06 1998 - loadConfig: initial:
     INFO: name "server-checker" reserved for owner
     "Agent Tcl administrator <cs.dartmouth.edu>"
   Tue Mar 10 12:19:06 1998 - loadConfig: initial:
     INFO: loading interpreter list
   Tue Mar 10 12:19:06 1998 - loadConfig: initial:
     INFO: adding "STATE-TCL" to interpreter list
   Tue Mar 10 12:19:06 1998 - loadConfig: initial:
     INFO: adding "STATE-TK" to interpreter list
   Tue Mar 10 12:19:06 1998 - loadConfig: initial:
     INFO: 3072 random bits read from file
     "/usr/agenttcl/access/random.bin"
   Tue Mar 10 12:19:06 1998 - agentd:
     INFO: server keyname is
     "Agent Tcl server <10.0.0.1> <grumpy.dartmouth.edu>"
   Tue Mar 10 12:19:06 1998 - loadConfig: initial:
     INFO: IP address of machine "localhost" is 127.0.0.1
   Tue Mar 10 12:19:06 1998 - loadConfig: initial:
     INFO: IP address of machine "grumpy.dartmouth.edu" is 10.0.0.1
   Tue Mar 10 12:19:06 1998 - loadConfig: initial:
     INFO: IP address of machine "localhost" is 127.0.0.1
   Tue Mar 10 12:19:06 1998 - loadConfig: initial:
     INFO: keyname for TrustedMachine "localhost" is
   ```

[1] The other set of installation instructions talks about these fields in more detail.

```
  "Agent Tcl server <127.0.0.1>"
Tue Mar 10 12:19:06 1998 - loadConfig: initial:
  INFO: IP address of machine "grumpy.dartmouth.edu" is 10.0.0.1
Tue Mar 10 12:19:06 1998 - loadConfig: initial:
  INFO: keyname for TrustedMachine "grumpy.dartmouth.edu" is
  "Agent Tcl server <10.0.0.1>"
Tue Mar 10 12:19:06 1998 - agentd:
  INFO: environment variable "HOME" is set to "/home/agenttcl"
Tue Mar 10 12:19:06 1998 - agentd:
  INFO: socketd has been started (process id 7543)
```

After the last line, you will be returned to the command prompt, since the server is now running as a daemon. All further messages are logged in the file *agent.log*, which is in the LOG_DIRECTORY that you specified in the Makefile.

To verify that the server is running as a daemon, check for *agentd* processes. On the author's machine:

```
% ps ax | grep agentd
32080 pp4 S     0:00 agentd -host bald.cs.dartmouth.edu ...
32081 pp4 S     0:00 agentd -host bald.cs.dartmouth.edu ...
%
```

The first *agentd* maintains the server's internal tables. The second *agentd* watches for incoming agents on the TCP/IP port. Both processes must be running for the server to function correctly. Check the screen output and the log file for error messages if one or both are missing. In the error log, the first *agentd* is referred to as "agentd", and the second *agentd* is referred to as "socketd", since it watches the TCP/IP and Unix domain sockets.

The server can be run in the foreground by editing the file *start_agentd* and adding the flag

```
-nodaemon
```

to the line that actually starts *agentd*. All error and informational messages will be displayed on the terminal rather than written to the log file. The foreground mode is useful for server development and intense debugging.

5. Edit the file *start_man* and add one of the following options to the line that actually starts up *manager.tcl*.

```
-nosecurity allow
```

```
-nosecurity allow-time
```

```
-nosecurity disallow
```

The option must be inserted after the *manager.conf* but before the ampersand. For example, after you are done editing, the last two lines of *start_man* might be

```
/usr/agenttcl/server/manager.tcl \
        /usr/agenttcl/access/manager.conf -nosecurity allow &
```

The first option causes the resource manager to grant all resource requests (resource requests come from agents executing on the local machine). The second option causes the resource manager to *reject* all resource requests except for requests for a longer lifetime. Finally, the third option causes the resource manager to reject all resource requests.

Note that *each of these options provides the same amount of security.* Since you are not using PGP, every agent can bypass the resource manager and access anything that the *dagents* user can access. This is the main reason that we created the special *dagents* account. Once you do install PGP, agents will *not* able to bypass the resource manager.

To reiterate, *since PGP is not installed, the resource manager has no security function.* Instead, it simply allows you to write your agents in the same way that you would if PGP were installed. Thus, since many of the example agents will not work unless the resource manager accepts all resource requests, I recommend that you use the first option, *-nosecurity allow.*

6. Start the resource manager by typing

   ```
   ./start_man
   ```

7. To bring down the server later, you need to kill all *agentd*, *agent*, and *agent-tk* processes (there will be quite a few if there are several agents executing on the local machine). Sending a SIGTERM to any *agentd* process will kill all *agentd* processes and most of the *agent* and *agent-tk* processes. The remaining *agent* and *agent-tk* processes must be killed individually.

8. If you change *agentd.conf*, you need to force the server to reload its configuration information. You can either (1) send a SIGHUP to any *agentd* process or (2) stop and restart the server as per the previous steps.

9. Finally, it is okay to run *start_agentd* from a system bootup script. The server will switch to the userid specified in the configuration file as soon as it starts executing.

4 Utilities

There are several useful utilities included with the system: *genrand*, *stirrand*, *seerand* and *machine.tcl*. The first three utilities, *genrand*, *stirrand* and *seerand*, are used to generate and stir the server's random bits file. These three utilities are in the SERV_INSTALL directory; the random bits file is the file *random.bin* in the DATA_DIRECTORY directory. To generate a new random bits file, use the *genrand* command.

```
genrand random.bin 3072
```

The second argument is the number of random bits to place in the file. Make sure that you use 3072 here, since the server will refuse to continue if less than 3072 random bits are available. To "stir" the random bits with the contents of an arbitrary file, use the *stirrand* command.

```
stirrand random.bin < filename
```

Finally, to see the random bits as human-readable hex numbers, use the *seerand* command.

```
seerand random.bin
```

You should not use the *seerand* command too often since it involves transmitting the random bits in "clear-text" at some point (e.g., between the remote disk and the local machine, between the local machine and the screen, etc.). You might use it occasionally, however, if you want to verify that the D'Agents developers have not done anything stupid and accidentally filled *random.bin* with zeros.

After using *stirrand* or *genrand*, you must tell the server to reload the random bits file. You can do this by sending a SIGHUP to any *agentd* process (or by stopping and restarting the server as described above). In addition to any regeneration and stirring that you do, the server writes out a new random.bin file whenever it is shutdown, and the server uses the types and arrival times of agent requests to continually stir the random pool as it executes.

The last utility, *machine.tcl*, is found in the EXEC_INSTALL directory. First, you must edit *machine.tcl* and change the four variables at the top to values that are appropriate for your site. In particular, the *machines* variable must list the machines on which *you* are running the D'Agents server. Now if you just type

```
machine.tcl
```

you will see a list of the machines on which the server *should* be running. If you type

```
machine.tcl all
```

instead, *machine.tcl* will *confirm* that the server is running on each machine by sending out child agents and waiting for responses. You will see informational messages as it submits the agents and receives the responses. Once *machine.tcl* has received all the responses or timed out, it will display a list of errors. Errors almost always indicate that the server is not running on the specified machine. If you want to check only specific machines, you can list those machines on the command line. For example,

```
machine.tcl m q
```

confirms that the server is running on machines m and q. Finally, if you want to get reports periodically as to which servers are down, you can set up

```
machine.tcl all
```

as a cron job that e-mails its output to a given user.