Unicornscan Documentation
Getting Started

Presented to End Users
Dec 20, 2007

Important Notice:
This documentation is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

UNICORNSCAN
Unicorns are fast! :)

Website:	http://www.unicornscan.org
SF Project:	http://www.sf.net/projects/osace
IRC Information:	#unicornscan of Efnet

Table of Contents

1 Installing Unicornscan **1**
- 1.1 Introduction . . . 1
  - 1.1.1 Requirements . . . 1
- 1.2 Installing the Prerequisites . . . 1
  - 1.2.1 PostgreSQL . . . 1
  - 1.2.2 libdnet . . . 2
  - 1.2.3 lib tool . . . 3
  - 1.2.4 libpcap . . . 3
- 1.3 Installing Unicornscan . . . 3
  - 1.3.1 Command-line and Front-End . . . 3
  - 1.3.2 Downloading Unicornscan . . . 3
- 1.4 Customized UNIX Compilation and installation from source code . . . 4
  - 1.4.1 Configure Directives . . . 4
- 1.5 Compilation Problems . . . 5
- 1.6 Getting the Front-End Working . . . 5

2 Examples **6**
- 2.1 Notes About Using the Tool . . . 6
- 2.2 Example Use . . . 6
  - 2.2.1 Basic TCP SYN Scan . . . 6
  - 2.2.2 UDP-Protocol-Specific-Payload Based Scanning . . . 6
  - 2.2.3 TCP Scanning . . . 7
  - 2.2.4 Saving to PCAP . . . 7
- 2.3 TCP - Various Flags . . . 8
- 2.4 Cheat Chart for NMAP Users . . . 9

CHAPTER 1
Installing Unicornscan

1.1 Introduction

Unicornscan can be installed from a package for your distribution. This guide is intended for those wishing to manually compile unicornscan for their specific needs.

Use the table of contents to skip directly sections that seem relevant to you. This guide describes how to install unicornscan on most POSIX platforms.

1.1.1 Requirements

Unicornscan is in constant development. We do our best to provide stable releases, but the code relies on specific libraries to be installed properly on the system. Many of the required libraries are included in the Unicornscan tar ball for convienience. However, if the configure script finds an older copy of the same library already installed on your system, it assumes that you intended it to use your previously installed version. This is especially frustrating with libraries such as libpcap.

Specifically, we are currently using the following libraries:

- libpq (for storing results, required for front-end - comes with PostgreSQL)
- libdnet-1.11
- libpcap-0.9.8

1.2 Installing the Prerequisites

Though we do try to bundle many of the prerequisite libraries with the tar ball, if you wish to install them system wide, follow the instructions below.

1.2.1 PostgreSQL

A very thorough documentation on installing and compiling PostgreSQL is available at:
http://www.postgresql.org/docs/8.2/interactive/installation.html

The quick installation is as follows:

1. Ensure that your OS/Distribution does not have an older version already installed. If it does, uninstall it.
2. Download the source:

```
$ wget ftp://ftp.postgresql.org/pub/source/v8.2.5/postgresql-8.2.5.tar.gz
```

3. Uncompress, change directories, configure, make

```
$ tar zxvf postgresql-8.2.5.tar.gz; cd postgresql-8.2.5; ./configure; make
```

4. As a priveldeged user, make install:

```
# make install
```

5. Add a postgres user to the system:

```
# adduser postgres
```

6. Create a directory to store the databases and change ownership to the postgres user:

```
# mkdir /usr/local/pgsql/data; chown postgres /usr/local/pgsql/data
```

7. Become the postgres user:

```
# su - postgres
```

8. As the postgres user, initialize the database and start the database service:

```
$ /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
$ /usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data >logfile 2>&1 &
```

9. Create a test database and connect to it to verify it is functioning properly:

```
$ /usr/local/pgsql/bin/createdb test
$ /usr/local/pgsql/bin/psql test
```

Once PostgreSQL is installed, don't worry about configuring additional users, databases, tables, etc. That will be documented later in the 1.6 (Getting the Front-End Working) section.

1.2.2 libdnet

1. Ensure that your OS/Distribution does not have an older version already installed. If it does, uninstall it.
2. Download the source:

```
$ wget http://easynews.dl.sourceforge.net/sourceforge/libdnet/libdnet-1.11.tar.gz
```

3. Uncompress, change directories, configure, make

```
$ tar zxvf libdnet-1.11.tar.gz; cd libdnet-1.11; ./configure; make
```

4. As a priveldeged user, make install:

```
# make install
```

1.2.3 lib tool

1. Ensure that your OS/Distribution does not have an older version already installed. If it does, uninstall it.
2. Download the source:

```
$ wget http://ftp.gnu.org/gnu/libtool/libtool-1.5.24.tar.gz
```

3. Uncompress, change directories, configure, make

```
$ tar zxvf libtool-1.5.24.tar.gz; cd libtool-1.5.24; ./configure; make
```

4. As a priveldeged user, make install:

```
# make install
```

1.2.4 libpcap

1. Ensure that your OS/Distribution does not have an older version already installed. If it does, uninstall it.
2. Download the source:

```
$ wget http://www.tcpdump.org/release/libpcap-0.9.8.tar.gz
```

3. Uncompress, change directories, configure, make

```
$ tar zxvf libpcap-0.9.8.tar.gz; cd libpcap-0.9.8; ./configure; make
```

4. As a priveldeged user, make install:

```
# make install
```

1.3 Installing Unicornscan

1.3.1 Command-line and Front-End

Unicornscan can be used solely from the command-line, but this guide will also detail how to use it with a PostgreSQL powered Front-End to get the most out of the data collected.

1.3.2 Downloading Unicornscan

unicornscan.org is the official source for downloading unicornscan source code and binaries for unicornscan. Source code is distributed in Gzip compressed tar files, and binaries are available for Linux (.tgz format). Find all of this at http://www.unicornscan.org/download.html.

1.4 Customized UNIX Compilation and installation from source code

Source installation is intended to be a painless process. The build system is designed to auto-detect as much as possible. Here are the steps required for a default install:

1. Download the 0.4.7 version of Unicornscan from http://www.unicornscan.org/

   ```
   $ wget http://www.unicornscan.org/releases/unicornscan-0.4.7.tar.bz2
   ```

2. Decompress the downloaded tarball with a command such as:

   ```
   $ tar jxvf unicornscan-0.4.7.tar.bz2
   ```

3. Change into the newly created directory:

   ```
   $ cd unicornscan-0.4.7
   ```

4. Configure the build system:

   ```
   $ ./configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var \
      --enable-bundled-ltdl --with-pgsql
   ```

   Run `./configure --help` for more information about the options shown (or read below). *Note*: The `--with-pgsql` option requires that you have the PostgreSql headers installed, and assumes you have a functional installation in place. If you need assistance installing, Postgres, see the section on installing prerequisite software.

5. Build Unicornscan:

   ```
   $ make
   ```

   Note that GNU Make is required. On BSD-derived UNIX systems, this is often installed as gmake. So if make returns a bunch of errors such as ”`Makefile, line 1: Need an operator`”, try running `gmake` instead.

6. As a privledged user, install Unicornscan, support files, docs, etc.:

   ```
   # make install
   ```

   Congratulations! Unicornscan is now installed as `/usr/bin/unicornscan`! Run it with the `-h` flag for a quick help screen.

7. To uninstall:

   ```
   # make uninstall
   ```

1.4.1 Configure Directives

Most of the UNIX build options are controlled by the configure script, as used in step number four above. There are dozens of command-line parameters and environmental variables which affect the way Unicornscan is built. Run `./configure --help` for a huge list with brief descriptions. Here are the ones that are specific to Unicornscan or particularly important:

`--prefix=directoryname`

This option, which is standard to the configure scripts of most software, determines where Unicornscan and its components are installed. By default, the prefix is `/usr/local`, meaning that unicornscan is installed in `/usr/local/bin`, the man page (unicornscan.1) is installed in `/usr/local/man/man1`, the data files (modules, payloads, etc.) are installed under `/usr/local/share/unicornscan`, and the configuration files are installed under `/usr/local/etc/unicornscan`.

If you only wish to change the path of certain components, use the options –bindir, –datadir, and/or –mandir.

1.5 Compilation Problems

If you run into trouble getting it compiled, feel free to jump onto IRC (efnet) on channel #unicornscan. Optionally you can also send an email to the OSACE mailing list, or try to install a precompilied binary package.

1.6 Getting the Front-End Working

To get the Front-End up and running, you will also need to install:

- PostgreSQL
- Apache
- mod_php

CHAPTER 2

Examples

2.1 Notes About Using the Tool

The basic functionality of Unicornscan is intended to be fairly simple to use. Help is available via the -h or –help options.

Unicornscan must be run as root. Shortly after execution, Unicornscan will change effective UID to nobody inside of a chrooted environment. This is done to limit inherent risks in taking arbitrary user input off of the wire.

Unicornscan can be used as a scalable port scanner. It uses CPU specific instructions to track the packets per second you specify as closely as possible. From a single Pentium system, it is typical to be able to generate up to 25,000 PPS or more (depending on hardware capabilities). The PPS limit will scale with your architecture accordingly. This single system PPS limit can be scaled however as we support clusters of scanners working together.

If you follow the OSSTMM (www.osstmm.org) 2.1 methodology for security testing, Section C covers Internet Technology Security testing. The very first step is Logistics and controls into the target network.—
For Unicornscan, proper logistics and controls involves calculating the bandwidth and packets per second appropriate for the network you are testing. With Unicornscan the -r option specifies the packet rate. A conservative number to start with is 100 PPS or -r100. Other common speeds used are 300, 500, 1000, etc. Warning, 10,000+ packets per second can crash many devices. Try to avoid scanning through a network device that peforms NAT, as our scans are likely to overrun their state table. Port scanning through NAT devices is not reliable and should be avoided where possible. To ensure accurate results, scan from a machine with as few networking devices in it’s way as possible. Also avoid using local state tracking or filtering software, such as firewalls.

2.2 Example Use

2.2.1 Basic TCP SYN Scan

```
# unicornscan server
TCP open                  domain[   53]         from 192.168.0.2  ttl 64
TCP open                    http[   80]         from 192.168.0.2  ttl 64
TCP open                   mysql[ 3306]         from 192.168.0.2  ttl 64
TCP open             xmpp-server[ 5269]         from 192.168.0.2  ttl 64
```

In this example, all that was specified is the name of a server we wanted to scan. The hostname server was resolved to the address of 192.168.0.2. A TCP SYN (-mTS, which is the default scan mode) scan was sent to that IP on the Unicornscan Quick Ports (default port list – same as server:q), as defined in the etc/unicornscan/unicorn.conf file. IP Addresses that respond with a SYN/ACK return as open.

2.2.2 UDP-Protocol-Specific-Payload Based Scanning

```
# unicornscan -r200 -mU -I 192.168.0.0/24:53
```

```
UDP open 192.168.0.2:53  ttl 64
UDP open                  domain[   53]         from 192.168.0.2  ttl 64
```

Option	Description
-r200	200 Packets Per Second
-mU	Scan Mode UDP
-I	Immediately display results to the screen as received
:53	Port 53
etc/unicornscan/payloads.conf	This file defines which static payloads are sent.

This command scans the 192.168.0.0/24 range for DNS servers. It sends DNS protocol specific queries as defined by etc/unicornscan/payloads.conf to UDP port 53 for every IP in the range. The IP's that respond return as open. To see all responses, specify the -E option. While scanning in UDP mode, a Closed signifies that we received an ICMP Type 3 code 3 packet from the IP being scanned. Any other ICMP response would be displayed with it's Type and Code listed in a T##C## format. Ex., if we got a Destination Unreachable, Communication Administratively Prohibited ICMP packet, it would be shown as T03C13. ***Note***: Any UDP packet destined for our source IP will show up as open. Ex, if you are web surfing from the same IP that is performing UDP scans, your DNS servers will show up as Open in your scan results. It is a good idea to scan from an IP that is not initiating any other traffic.

2.2.3 TCP Scanning

```
# unicornscan -r500 -mT www.yahoo.com/29:80,443
TCP open                    http[   80]         from 87.248.113.8  ttl 47
TCP open                   https[  443]         from 87.248.113.8  ttl 47
TCP open                    http[   80]         from 87.248.113.9  ttl 46
TCP open                   https[  443]         from 87.248.113.9  ttl 46
TCP open                    http[   80]         from 87.248.113.10  ttl 46
TCP open                   https[  443]         from 87.248.113.10  ttl 46
TCP open                    http[   80]         from 87.248.113.11  ttl 46
TCP open                   https[  443]         from 87.248.113.11  ttl 46
TCP open                    http[   80]         from 87.248.113.12  ttl 46
TCP open                   https[  443]         from 87.248.113.12  ttl 46
TCP open                    http[   80]         from 87.248.113.13  ttl 46
TCP open                   https[  443]         from 87.248.113.13  ttl 46
TCP open                    http[   80]         from 87.248.113.14  ttl 47
TCP open                   https[  443]         from 87.248.113.14  ttl 47
TCP open                    http[   80]         from 87.248.113.15  ttl 47
```

Option	Description
-r500	200 Packets Per Second
-mT	Scan Mode TCP (TCP is default mode if not otherwise specified)
:80,443	Ports 80 and 443

This command resolves the name www.yahoo.com to 87.248.113.14 and sends packets to the IP range of 87.248.113.14/29 (87.248.113.8-16) on TCP ports 80 and 443.

2.2.4 Saving to PCAP

```
# unicornscan 10.23.0.0/22:161 -r1000 -I -v -mU -R3 -P "not port 162" \
  -w snmp.pcap -s 10.23.0.1
```

Option	Description
-v	Slightly more verbose output (can be used multiple times, Ex. -vvv)
-P ”not port 162”	Pcap filter (man tcpdump)
-w snmp.pcap	Write output in pcap format to the file snmp.pcap
-R 3	Resend the probe attempt 3 times
-s 10.23.0.1	Send the packets from the source IP address of 10.23.0.1
-W 6	Send the packets with the linux OS personality

The -P option is used to control additional pcap filters. For example if you wanted to exclude responses from the host 192.168.5.16, you would add ’-P ”not host 192.168.5.16”’.

The -R option allows you to specify how many times to repeat sending a packet to the target range. Ex. -R3 will provide for 3 scans of the target range. Although the packets will go out 3 times, and you will likely get responses 3 times, the output will only show you the response once. Again, you can check the pcap file you wrote to with the -w option to validate what is received.

We included a -w option for unicornscan which works much like the -w option from tcpdump. This allows you to write all of the response data collected to a pcap file to parse later if you need to.

Different kernels have different implementations of the TCP/IP stack. These differences make up the ”fingerprint” characteristics that tools like p0f, ids, firewalls, etc look at to determine the OS that created the packet. Because we are using Raw Sockets, we can control these characteristics. We included seven OS simulations in the 0.4.7 release - Cisco, OpenBSD, WindowsXP, p0fsendsyn, FreeBSD, nmap, linux, and Crazy lint tcp header (use with p0f hopefully). You can specify which one to send as with the -W option.

2.3 TCP - Various Flags

We included the ability in the TCP scan mode to specify wich flags to enable. For example ‘-mTsFpU’ would enable SYN,FIN,No Push,URG bits inside the tcp header.
Quick list of common Flag schemes:
If you want a SYN scan -mT
If you want an ACK scan -mTsA
If you want an Fin scan -mTsF
If you want a Null scan -mTs
If you want a nmap style Xmas scan -mTsFPU
If you want a scan with all options on -mTFSRPAUEC
If you want to create your own special type of scan, just play with the
options and document the results.
(see http://www.iana.org/assignments/tcp-header-flags for more info)

2.4 Cheat Chart for NMAP Users

Scan Type	**nmap**	**unicornscan**
Syn Scan	-sS -v	(-mT) -Iv
Connect Scan	-sT -v	-msf -Iv
Syn + osdetect	-sS -O -v	-eosdetect -Iv (-mT)
UDP scan	-sU -v	-mU -Iv
IP Protocol Scan	-sO -v	NONE
FIN scan	-sF -v	-mTsF -v -E
NULL scan	-sN -v	-mTs -v -E
XMAS scan	-sX -v	-mTsFPU -v -E
ACK scan	-sA -v	-mTsA -v -E
scan ports 1 and 5	-sS -p1,5 -v	(-mT) host:1,5
scan ports 1 through 5	-sS -p1-5	(-mT) host:1-5
scan ALL tcp ports	-sS -p0-65535 -v	(-mT) host:a