MINDS
Minnesota Intrusion Detection System
User Manual

March 8, 2009

MINDS Group[*†]

[*]Department of Computer Science, University of Minnesota
[†]Army High Performance Computing and Research Center (AHPCRC), Minnesota

Contents

1 Introduction to MINDS

The *Minnesota INtrusion Detection System* (MINDS[1]) is an end-to-end data mining based intrusion detection solution that has been shown to be effective in detecting cyber-intrusions on large scale networks [1, 2].

1.1 Outline of MINDS's manual

The outline of this manual is as follows. Section 2 describes the installation process for MINDS and the system requirements. Section 3 describes how to execute MINDS from end to end and how to configure different parameters.

2 Installing MINDS

MINDS source code as well as binaries are available. The installation currently handles three platforms - Linux, Sun Solaris OS and FreeBSD.

2.1 System Requirements

MINDS is written in GNU C++ and tested extensively on Linux, Sun Solaris and FreeBSD. The distribution is available as source as well as binaries which can run on ix86 architectures.

2.2 External Libraries

The installation requires one of the two external pre-compiled libraries depending on the format of the input data (*Cisco Netflows*[2] or *tcpdump*[3] data). The installation can be carried out for either one of the formats or for both formats. Additionally, MINDS also requires an XML parser library[4] to parse the configuration files. The three libraries are also packaged with the MINDS distribution.

2.3 *install*

To install the software first copy the files to a local directory - *$local-home*. By default, MINDS will be installed in /usr/local, which requires root privileges. To install in a different directory, the directory name should be provided as $install-dir when installing the various components below.
Untar the files

> *tar -zxvf minds_0.1.tgz*

[1] http://www.cs.umn.edu/research/MINDS
[2] ftp://ftp.eng.oar.net/pub/flow-tools/flow-tools-0.66.tar.gz
[3] http://www.tcpdump.org/release/libpcap-0.8.3.tar.gz
[4] http://sourceforge.net/projects/tinyxml

To install *flow-tools*
> *cd $local-home/minds/src/flow-tools-0.68.2-rc5/*
> *./configure –prefix $install-dir*
> *make install*

To install *pcap*
> *cd $local-home/minds/src/libpcap-0.8.1/*
> *./configure –prefix $install-dir*
> *make install*

To install *xmlparser*
> *cd $local-home/minds/src/xmlparser/*
> *./configure –prefix $install-dir*
> *make install*
To install MINDS
> *cd $local-home/minds/*
> *./configure –prefix $install-dir*
> *make install*

3 Executing MINDS from End to End

This section describes how to run MINDS from end-to-end. The entire execution is carried out as a set of a number of steps which are individually described in the subsequent sections. The *minds* code's end to end execution comprises of following steps:

- *Conversion to mmr format* [No configuration parameters]
- *Merge and match uni-directional flows to bi-directional sessions* [mergematch-config.xml]
- *Scan detection and labeling* [scandetector-config.xml, blk.ports, dark.hosts, p2p.ports, p2p.hosts]
- *P2P detection* [p2pdetector-config.xml]
- *Behavioral anomaly detection and labeling* [anomalydetector-config.xml, minds.rules]

Each of these steps will be described in detail in subsequent sections below.

3.1 Setting Configuration Parameters

As mentioned earlier, the MINDS 2.0 system involves multiple execution steps. The output of each step can be controlled by specifying several threshold and other control parameters. Moreover, several components require knowledge of the internal network structure and other domain related information. All these are specified using a set of configuration files. The description of the configuration files for each of the section are

given below. The configuration files are typically in XML format. Default versions of configurations files are provided in *$install-dir/etc/cfg*.

3.1.1 *mergematch-config.xml*

This file is required by the Merge and Match routine to specify the internal IP addresses. Multiple internal subnetworks and the corresponding masks can be specified.

3.1.2 *scandetector-config.xml*

This file is required by the Scan Detector to specify the internal IP addresses. Multiple internal subnetworks can be specified. Additionally, several parameters can be specified for scan detection.

3.1.3 *blk.ports*

List of blocked ports for the target network. Each entry should be specified per line (separated by carriage return) by the protocol followed by a space followed by the port number. For e.g.
TCP 1993

3.1.4 *dark.hosts*

List of dark IPs inside the target network. Each line should contain the IP (in dot format) of a host. For e.g.
198.162.222.2

3.1.5 *p2p.ports*

List of known p2p ports inside the target network. Each entry should be specified per line (separated by carriage return) by the protocol followed by a space followed by the port number. For e.g.
TCP 41170

3.1.6 *p2p.hosts*

List of known p2p hosts inside the target network. Each line should contain the IP (in dot format) of a host. For e.g.
198.162.222.2

3.1.7 *p2pdetector-config.xml*

This file is required by the P2P Detector to specify various parameters as well as list of known p2p/malware/good ports.

3.1.8 *anomalydetector-config.xml*

This file contains the configuration parameters which can be set by the user to run the anomaly detector.

3.1.9 *minds.rules*

This file describes how the rule files can be used to filter the data.

- *Ruleset* keyword can be used to combine multiple runs in one shot. Anomaly detection is run for every subset of flows corresponding to each ruleset.
- After a Ruleset keyword, rules can be typed in. There are two types of rules: *select* and *ignore*. The default action of a ruleset is ignore all, i.e. if no select rule applies for a given record, it's ignored. Not all the rules have to be executed for every single flow record. The action suggested by the rule (select / ignore) is applied right away when a rule matches the record, i.e. if a select rule matches the record, it's added to the subset even if a later ignore rule matches the record too. The precedence of the rules is from top to bottom; if the first rule doesn't apply, only then the second will be applied.
- After a select / ignore keyword, one of the following keywords can be used. *srcip, dstip, srcport, dstport, protocol, packets, octets or all*. The operations that can be specified on these fields are: >=, >, ==, !=, <=, <, *inside, outside, net_equal, net_not_equal*. The following keywords *scans, p2p, hdt* can be used after the select / ignore keyword for flows labeled as scans, p2p or high port data transfers.
- The operations should be followed by values of appropriate type. Multiple rules on one line will be interpreted as AND'ed together. "inside" and "outside" can be used provided that the boundaries of the network should be specified in the config file.

In the case of 'all', the rule will match anything and the action suggested by the rule will be executed right away. Table 1 shows some sample rules.

(a)
ruleset example
  ignore srcIP inside dstIP inside
  ignore srcIP outside dstIP outside
  select srcport > 1024 dstport > 1024 protocol == 6
  ignore protocol == 17
  select srcip >= 1.1.1.0 srcip <= 1.1.1.255
  select srcip net_equal 1.1.1.0 24
  select srcip net_equal 1.1.1.0 255.255.255.0

(b)
ruleset all
  select all

Table 1: Two sample rulesets for MINDS Anomaly Detection Component. The last three lines of rule-set example are equivalent.

4 MINDS Data Format - *mmr*(merged and matched record)

This is the MINDS internal data format. The input data is transformed to this format using corresponding converters. The *mmr* data structure has following fields:

1. start timestamp
2. end timestamp
3. creation timestamp
4. source ip
5. source port
6. destination ip
7. destination port
8. number of packets sent
9. number of packets received
10. number of bytes sent
11. number of bytes received
12. protocol
13. TCP flags
14. source mask
15. destination mask
16. source AS number
17. source AS number
18. scan bit // NOT SCAN, SCAN WITHOUT REPLY, SCAN WITH REPLY
19. scan score // score upon declaration (scanner/benign)
20. scan reason // RSN UNKNOWN, KNOWNSCANNER, SCORE, BLOCKED, EMPTY, ERASESCANNER, OLDESTRECORD, OLDESTSCORE
21. p2p bit // NOT_P2P, P2P
22. high port data transfer bit // NOT_HPDT, HPDT
23. lof_anomaly_score

4.1 Output Formats

All MINDS components (except for the format converters) operate on the binary files containing mmr flows and output binary mmr files with certain bits modified. MINDS anomaly detector generates two additional outputs. The first output are the top anomalous flows, as determined by the anomaly detector, in text format. The second output is the summarized output of the top anomalous flows in text format.

The annotated output file of MINDS is a text file with each line corresponding to suspicious connection. Each line has 38 attributes. The labels and description of these attributes is given in table 2.

Column	Label	Description
1	Connection ID	*ID of the Connection*
2	Anomaly Score	*Score assigned by Anomaly Detector*
3	Time Stamp	*Time at which the connection starts*
4	duration	*Duration in seconds for which the connection lasted*
5	Src IP/ Src Port	*Source IP and the Source Port in the connection*
6	Dst IP/ Dst Port	*Destination IP and the Destination Port in the connection*
7	Protocol	*Protocol - tcp, udp, icmp, arp etc.*
8	ttl	*Time to live - Defined for TCP connections*
9	TCP Flags	*Defined for TCP Connections*
10	window size	*Defined for TCP Connections*
11	packets sent	*Number of packets sent from src to dst*
12	bytes sent	*Number of bytes sent from src to dst*
13	packets received	*Number of packets sent from dst to src*
14	bytes received	*Number of bytes sent from dst to src*
15	p2p bit	*0 - normal connection, 1 - p2p connection*
16	scan bit	*0 - normal connection 1- scan 2 - scan with a reply*
17	hpdt bit	*0 - normal connection 1- high random port data transfer*
18	inside bit	*0 - dst ip inside network 1 - src ip inside network*
19-38	Contribution Vector	*Assigned by the Anomaly Detector*

Table 2: Description of each column in the final output of MINDS

4.2 *nf2mmrecord*

Converts Cisco Netflow data to mmr format.
Usage: cat <input-file> | ./nf2mmrecord -o <output-file>

- *input-file* - Location of the binary file containing Cisco Netflows.
- *output-file* - mmr file where the output mmr data will be stored.

4.3 *collector*

Converts TCPDump data to intermediate flow format. **Usage**: collector [-i <input_tcpdump_filename> | -e <interface_name>] -o <output_filename>

- *input_tcpdump_filename* - Location of the input tcpdump file.

- *interface_name* - Name of the interface from which the packets are to be captured. Note that only one of the above two options can be used to specify the input source.
- *output-file* - tcpdump intermediate flow file name where the output flows are stored.

4.4 *e2mmrecord*

Converts the tcpdump intermediate flows to mmr format.
Usage: ./e2mmrecord -i <input-file> -o <output-file>

- *input-file* - Location of the binary file containing tcpdump intermediate flows.
- *output-file* - mmr file where the output mmr data will be stored.

4.5 *mmrcat*

Prints the mmr flows in text format onto stdout.
Usage: ./mmrcat -i <input-file>

- *input-file* - Location of the input binary mmr file.

4.6 *merge-match*

Merges and matches the uni-directional flows into bi-directional sessions.
Usage: ./merge-match -i <input-file> -o <output-file>

- *input-file* - Location of the input binary mmr file.
- *output-file* - mmr file where the output mmr data will be stored.

5 Scan Detector and Labeler - *scan_detector*

Detects and labels connections based on their scan like behavior. The scan detector assigns different labels to the connections and a labeler classifies the connections as *normal*, *scans*, *p2p* or *high port data transfer*. Note that the scan detector assigns labels only to outgoing connections from the internal network(s) as specified in the configuration file.
Usage: ./scan_detector <input-file><config-file><blocked ports><dark IPs><known p2p ports><known p2p hosts><output–file>

- *input-file* - Location of the input binary mmr file.

- *config-file* - Location of the XML configuration file (see section 3.1.2).
- *blocked ports* - Location of file containing list of blocked ports (see section 3.1.3)
- *dark hosts* - Location of file containing list of dark IPs inside the network (see section 3.1.4)
- *p2p ports* - Location of file containing list of known p2p ports (see section 3.1.5)
- *p2p hosts* - Location of file containing list of known p2p hosts (see section 3.1.6)
- *output-file* - mmr file where the output mmr data with scan labels will be stored.

6 P2P Detector and Labeler - *p2p_detector*

Detect and labels p2p connections.

7 MINDS Anomaly Detector - *anomaly_detector*

Assigns anomaly scores to the connections based on their *lof* (local outlier factor) score. The output is the connections in a text format annotated with their anomaly scores and the contribution vector for different features.
Usage: ./anomaly_detector -i <input-file> -c <config-file> -r <rules-file> [-t <training-file>] [-n <number_of_threads>] -o <output-file>

- *input-file* - Location of the input binary mmr file.
- *config-file* - Location of the configuration file for the anomaly detector in XML format (see section **??**).
- *rules-file* - Location of the rules file for the anomaly detector (see section 3.1.9).
- *training-file* - Location of the file containing training data [optional parameter]. If this parameter is specified, the program reads training data samples from this file. If the number of samples read is less than the training data size specified in *config-file*, the remaining samples are randomly selected from the data. After anomaly detection the training data is written back to the *training-file*.
- *number_of_threads* - Number of threads to be run simultaneously [optional parameter].
- *output-file* - text file where the output annotated data will be stored. The format of the output file is shown in 4.1.

References

[1] V. Chandola, E. Eilertson, L. Ertoz, G. Simon, and V. Kumar. Data mining for cyber security. In A. Singhal, editor, *Data Warehousing and Data Mining Techniques for Computer Security*. Springer, 2006.

[2] L. Ertoz, E. Eilertson, A. Lazarevic, P.-N. Tan, V. Kumar, J. Srivastava, and P. Dokas. MINDS - Minnesota Intrusion Detection System. In *Data Mining - Next Generation Challenges and Future Directions*. MIT Press, 2004.