Intrusion Detection using a Rule-Based Classifier

Randeep S. Gakhal, BASc

CMPT 740, December 2003

Overview

- Introduction
- Intrusion Detection
- The RIPPER Classifier
- The 1998 DARPA Dataset
- Intrusion Detection Results
- Conclusions

Introduction

Introduction

- Goal:
  – Develop and evaluate a system that detects computer network intrusions

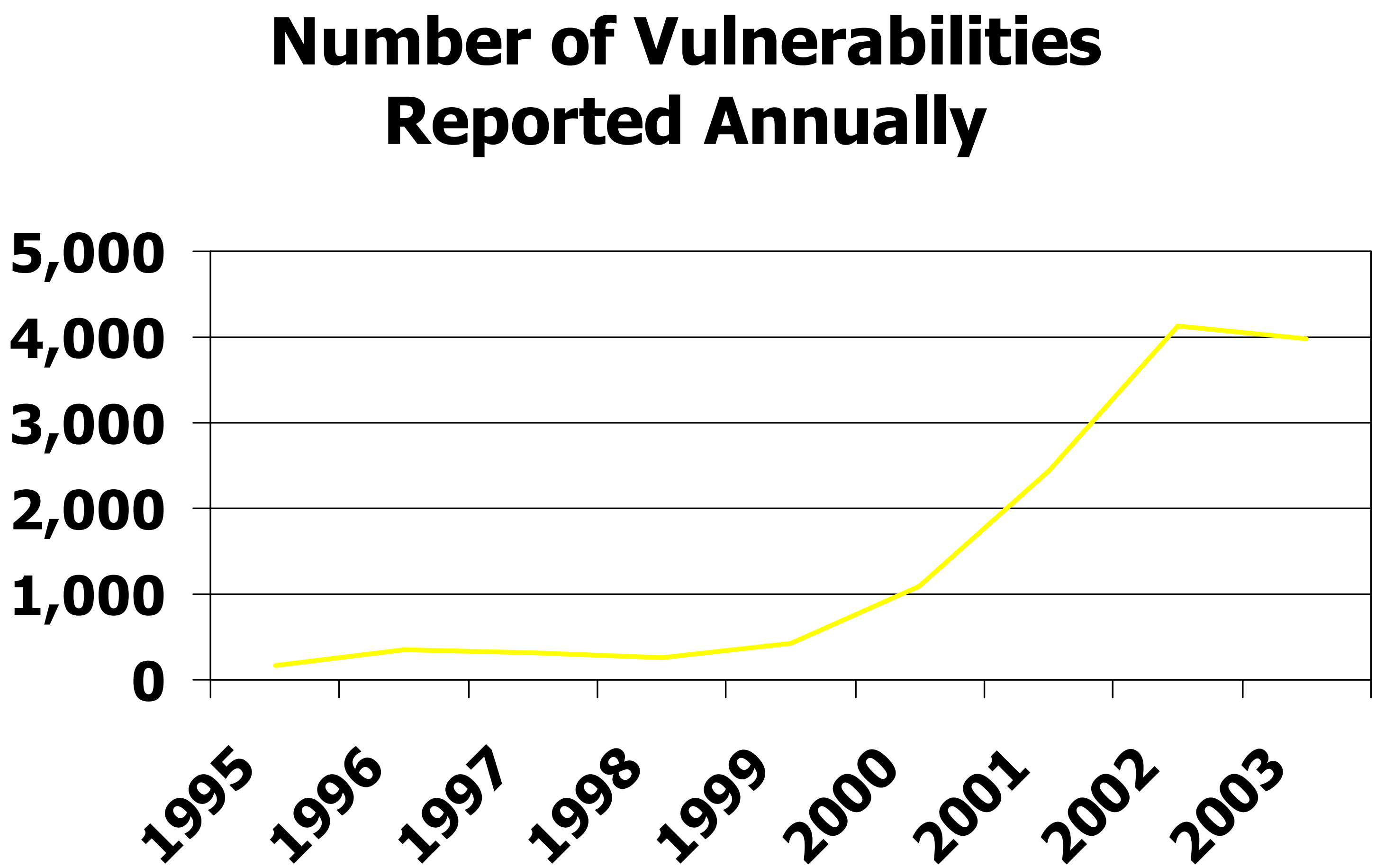
  → Intrusion Detection System
- We will be using a paradigm called "misuse detection" to detect intrusions

Intrusion Detection

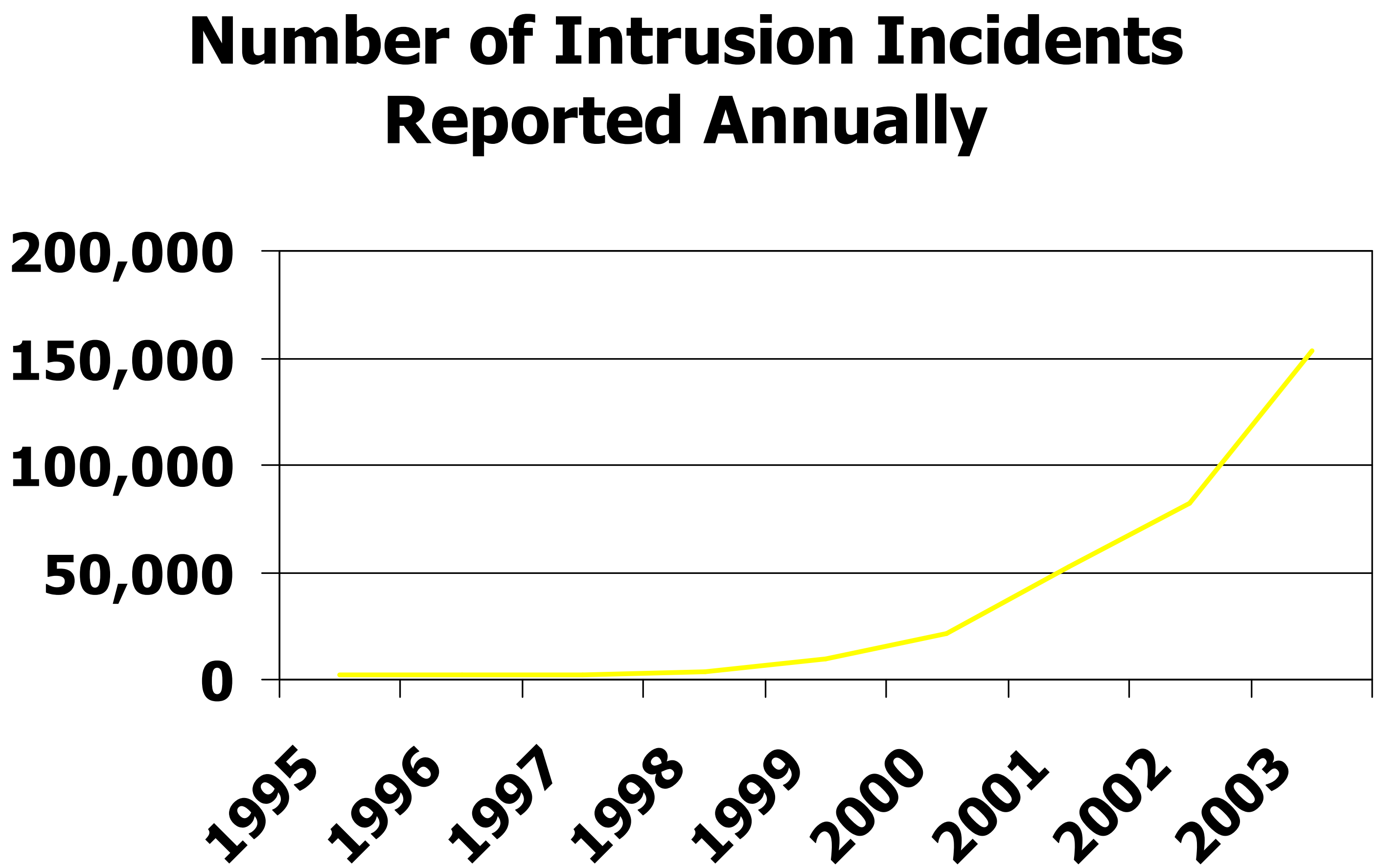
Intrusion Detection [1]

- Intrusions:
  - Actions that attempt to bypass security mechanisms of computer systems.
- Attacks originate from:
  - Users on the Internet accessing the system
  - Insiders trying to gain and abuse non-authorized privileges

Intrusion Detection [2]



Intrusion Detection [2]



Intrusion Detection

- Detecting intrusions requires monitoring large volumes of data - data mining makes intelligent detection possible.
- Two major techniques of intrusion detection employ data mining:
  – Misuse Detection
  – Anomaly Detection

Misuse Detection

- Record and learn patterns that represent an intrusion
- Monitor network traffic and detect intrusions based on the learned patterns
- Pro: Accurate at detecting learned intrusions
- Con: Limited to learned intrusions
  - NOT adaptive

Anomaly Detection

- Build a profile of typical network traffic over some attack free training period
- Monitor deviations from this profile on live traffic
- Pro: Can detect unknown intrusions
  - Adaptive
- Con: Statistics can be slowly trained so that an attack can go through undetected
- Con: Not suited for attacks that consist of a few connections

The RIPPER Classifier

The RIPPER Classifier [3]

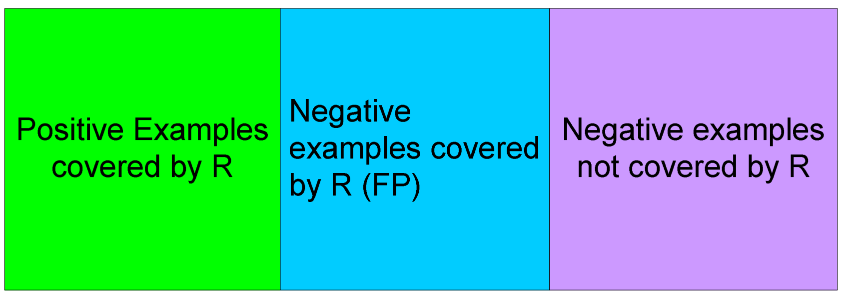
- Generates a series of classifier rules:
  - Eg: { Service = ICMP Echo Request;
    # conn’s in last 2 sec >= 5; }
    → SMURF attack
- Rules generated for each value of the target class
- Easy to read and check for “sanity” by a human

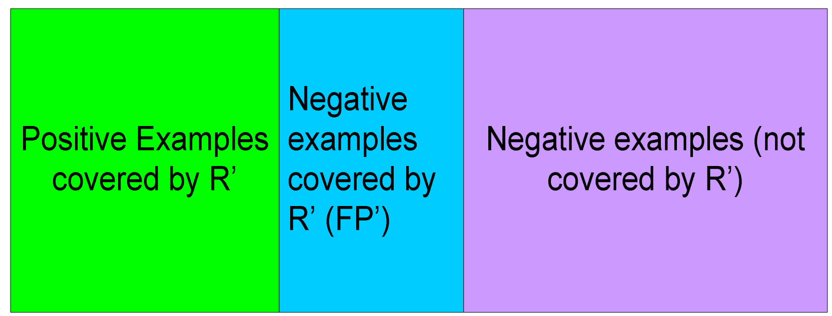
The RIPPER Classifier

- Training data randomly divided into a Growing Set and a Pruning Set
  - Ratio = ~2:1
- Repeatedly create rules in two phases:
  - Growing phase → Pruning phase
- Create rules for each class value in order of increasing prevalence

Growing Rules

- Rules are grown by adding conjectures that maximize information gain on Growing Set
- E.g., consider growing a rule R:





- We add the conjecture that maximizes DL-DL'

Pruning Rules

- A rule R is grown until no further information gain is possible. It is then pruned using the Prune Set.
- Conditions are removed from the rule, trying to maximize function:

$$\frac{p+(N-n)}{P+N}$$

- Where:
  - P is number of positive examples in Prune Set
  - N is number of negative examples in Prune Set
  - p is number of positive examples covered by R
  - n is number of negative examples covered by R

The 1998 DARPA Dataset

The 1998 DARPA Dataset[4]

- Lincoln Labs at MIT maintains datasets for testing intrusion detection systems
- DARPA 1998 dataset consists of:
  - 7 weeks of training data (~5M connections!)
  - 2 weeks of test data
- Data comprised of binary tcpdump data
- Comes with a preprocessed connection profile in text format

  → All other features have to be extracted yourself from the binary data

The 1998 DARPA Dataset

- 4 types of attacks are present:
  - Denial of Service (DOS)
    - Eg. ping-of-death, syn flood
  - Unauthorized access (R2L)
    - Eg. guessing password
  - User abuse of privileges (U2R)
    - Eg. buffer overflow attacks
  - Probing and surveillance
    - Eg. port scans

The KDD CUP 1999 Dataset [5]

- An “easier to digest” version of the DARPA 1998 dataset
- Binary tcpdump data has been intelligently processed to construct additional features

- Saved me a few months of work !!! ☺

The KDD CUP 1999 Dataset

- 3 classes of features:
  - Basic features: src, dst, service, duration, src bytes, dst bytes…
  - Content features: failed logins, # shells, su attempts…
  - 2 sec window features: conn count, SYN err rate, REJ err rate…
- Time window features allow our misuse detection approach to capture attacks better suited for anomaly detection → capture temporal dependencies

The KDD CUP 1999 Dataset

- Consists of:
  - Seven weeks of training data as one text file
    - ~750 Mb !
    - Attack patterns are the same
  - A 10% subset of training data
    - Contains instances of all attacks
    - Much easier to work with because of smaller size
      → No seg faults from running out of memory during training !
  - Two weeks of test data as another text file
    - Statistics and patterns of attacks have changed
    - Contains some new attacks

Intrusion Detection Results

Trials on DARPA 1998 Data Set

- Tried to train RIPPER using connection profile data provided on one or two days from the training set
- The resulting rules gave:
  - 100% accuracy on the data I trained on
  - 0% accuracy on everything else
- The rules completely overfit the data:
  - Connection profile did not give enough features to identify the true nature of an attack
  - Need more instances of an attack to develop more general features

Trials on KDDCUP 1999 Dataset

- Ran RIPPER on 10% training data file and obtained rules that were general and intuitive:
  - { Service = TELNET; Duration >= 299; Duration <= 337; Count >= 255; }
    - → SPY attack
  - { Failed logins >= 1; Same service rate >= 1; }
    - → Guess password attack

Trials on KDDCUP 1999 Dataset

- Application of rules to 7 weeks of training data:

	Number	Rate
Total Connections:	4,898,431	
Number Attacks:	3,925,650	
Correctly identified attacks:	3,925,190	99.99%
False positives:	284	0.03%
False negatives:	316	0.01%

Trials on KDDCUP 1999 Dataset

- Application of rules to 2 weeks of test data:

	Number	Rate
Total Connections:	311,029	
Number Attacks:	250,436	
Correctly identified attacks:	225,939	90.22%
False positives:	301	0.50%
False negatives:	21,256	8.49%

Trials on KDDCUP 1999 Dataset

- RIPPER was a success
- Results were in accordance with the paradigm of misuse detection:
  - Extremely high accuracy for instances with same pattern as those we trained on
  - Accuracy diminishes for attack instances with changing patterns
    - False negatives – from new attacks and evolved known attacks

Confusion Matrix – Training Data

	Predicted Class: normal	back	buffer_overflow	ftp_write	guess_passwd	imap	ipsweep	land	loadmodule	multihop	neptune	nmap	perl	phf	pod	portsweep	rootkit	satan	smurf	spy	teardrop	warezclient	warezmaster	False Pos
Actual Class: normal	972497	16	15	1	1	3	20	7	7	25	37	5	2		7	5	10	38	1	6		64	14	284
back	0	2203	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
buffer_overflow	0	0	28	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	
ftp_write	1	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
guess_passwd	0	0	0	0	53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
imap	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ipsweep	15	0	0	0	0	0	12432	0	0	0	0	32	0	0	0	0	0	0	0	0	0	2	0	
land	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
loadmodule	2	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
multihop	1	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	
neptune	15	0	0	0	0	0	0	0	0	0	1071988	0	0	0	0	10	0	4	0	0	0	0	0	
nmap	113	0	0	0	0	1	18	0	0	0	0	2184	0	0	0	0	0	0	0	0	0	0	0	
perl	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	
phf	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	
pod	0	0	0	0	0	0	0	0	0	0	0	0	0	0	264	0	0	0	0	0	0	0	0	
portsweep	21	0	0	0	0	7	10	0	0	0	20	2	0	0	0	10323	0	26	0	0	0	4	0	
rootkit	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	
satan	93	0	0	0	0	0	2	0	0	0	2	0	0	0	0	1	0	15794	0	0	0	0	0	
smurf	46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2807840	0	0	0	0	
spy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	
teardrop	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	979	0	0	
warezclient	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1019	0	
warezmaster	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	
False Neg	316																							

- Correctly identified (green)
- Attack successfully detected, however, incorrect attack predicted. (light cyan)
- False negatives. (red)
- False positives (yellow)

Confusion Matrix – Test Data

Predicted Class (columns) / Actual Class (rows)

	normal	apache2	back	buffer_overflow	ftp_write	guess_passwd	http_tunnel	imap	ipsweep	land	loadmodule	mailbomb	mscan	multihop	named	neptune	nmap	perl	phf	pod	portsweep	processtable	ps	rootkit	saint	satan	sendmail	smurf	snmpgetattack	snmpguess	spy	sqlattack	teardrop	udpstorm	warezclient	warezmaster	worm	xlock	xsnoop	xterm
normal	60292	0	6	0	0	0	0	0	2	0	1	0	0	0	0	14	0	0	0	16	65	0	0	0	0	152	0	0	0	0	0	0	39	0	4	2	0	0	0	0
apache2	51	0	469	0	0	0	0	0	0	0	0	0	0	0	0	274	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
back	0	0	1098	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
buffer_overflow	7	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ftp_write	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
guess_passwd	4063	0	1	0	0	303		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
http_tunnel	138	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
imap	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ipsweep	3	0	0	0	0	0	0	0	299	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0
land	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
loadmodule	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mailbomb	4983	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mscan	297	0	0	0	0	0	0	283	0	0	0	0	0	0	0	114	0	0	0	0	21	0	0	0		338	0	0	0	0	0	0	0	0	0	0	0	0	0	0
multihop	14	0	0	1	0	2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
named	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
neptune	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	57956	0	0	0	0	1	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0
nmap	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	84	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
perl	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
phf	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pod	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	87	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
portsweep	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	349	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
processtable	472	0	0	0	0	0	0	0	0	0	0	0	0	0	0	275	0	0	0	0	3	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ps	8	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rootkit	12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
saint	109	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	625	0	0	0	0	0	0	0	0	0	0	0	0	0	0
satan	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1628	0	0	0	0	0	0	0	0	0	0	0	0	0	0
sendmail	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smurf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	164091	0	0	0	0	0	0	0	0	0	0	0	0
snmpgetattack	7741	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
snmpguess	2406	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
spy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
sqlattack	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
teardrop	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0
udpstorm	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
warezclient	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1019	0	0	0	0	0
warezmaster	889	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	704	0	0	0	0
worm	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
xlock	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
xsnoop	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
xterm	4	0	0	6	0	0	0	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

False Neg 21257

- Correctly identified
- Attack successfully detected, however, incorrect attack predicted.
- False negatives.
- False positives
- New attack - not present in 7 week training data

Conclusions

Conclusions

- Misuse detection clearly excels at detecting known intrusion patterns
- Accuracy diminishes as attacks mutate
- Winner of KDDCUP 1999 had detection rate of 96% on test data (I had 90%)
  - →Lots of room for improvement
- Base RIPPER algorithm is extremely powerful

Misuse Detection (MD) vs. Anomaly Detection (AD)

- MD is more apt at handling real-time data than anomaly detection
  - → Almost all commercial systems use MD
- MD can detect attacks based on temporal statistics by constructing additional features
- MD Does not handle changing attacks well
  - → System can easily be retrained

Future Research

- Try other classification approaches
  - C5, FOIL, neural networks, k-nearest-neighbor…
- Combine anomaly detection and misuse detection
  - Anomaly detection can be used to detect when RIPPER rules need to be re-trained
- Distributed IDS?
  - Who maintains the misuse detection database in a network and how is it shared?

Questions?

References

1. A. Lazarevic et al. *Data Mining for Computer Security Applications*, IEEE ICDM 2003 Tutorial.
2. http://www.cert.org/stats/cert_stats.html
3. W. W. Cohen. *Fast Effective Rule Induction*, In Machine Learning: Proceedings of the Twelfth International Conference, Lake Tahoe, California, 1995.
4. http://www.ll.mit.edu/SST/ideval/data/data_index.html
5. http://www.kdnuggets.com/datasets/kddcup.html#1999