Using Key-String Selection and Neural Networks to Reduce False Alarms and Detect New Attacks with Sniffer-Based Intrusion Detection Systems

Richard P. Lippmann and Robert K. Cunningham
rpl@sst.ll.mit.edu

MIT Lincoln Laboratory
Room S4-121
244 Wood Street
Lexington, MA 02173-0073

Presented at the Recent Advances in Intrusion Detection, RAID 99 Conference,

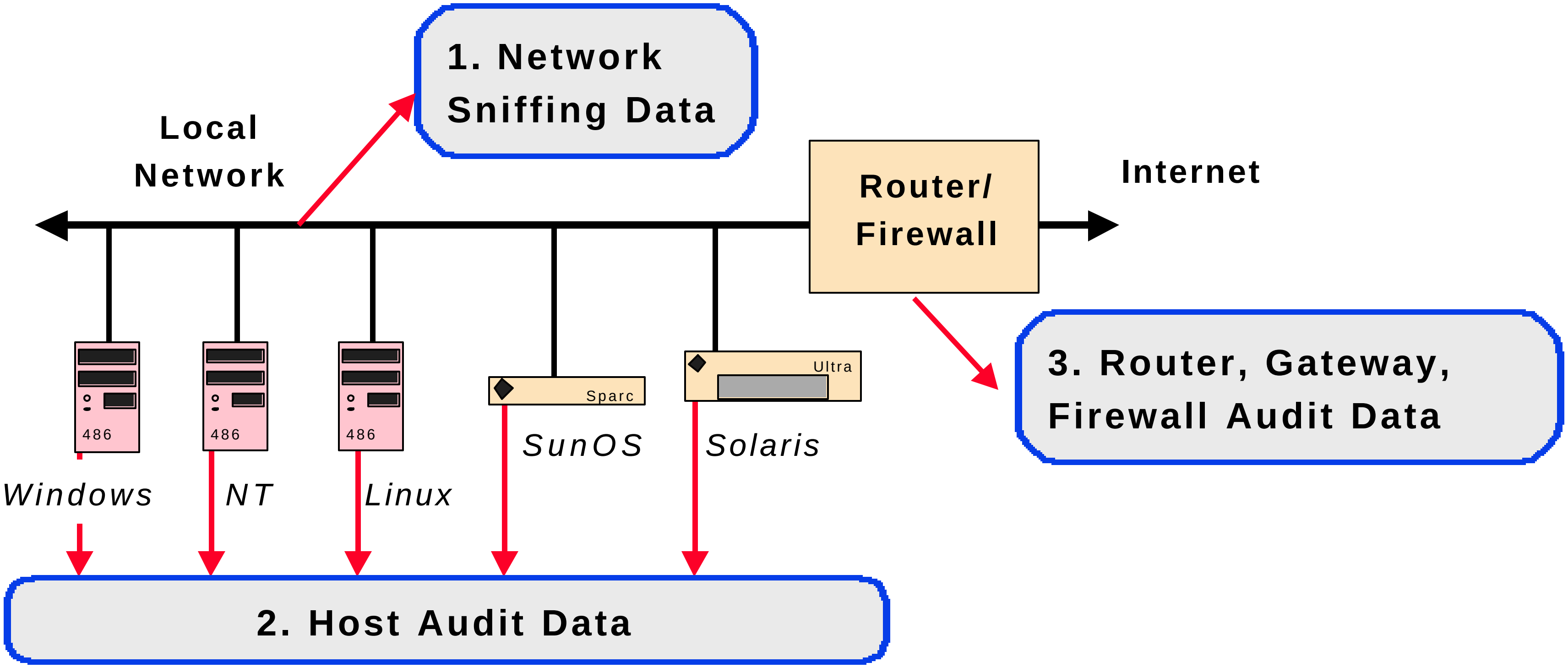
7-9 September

West Lafayette, Indiana, USA

Outline

- Intrusion Detection Background
- Goals and System Design
- Evaluation and Results
- Summary
- Future Work

Common Input Features for Intrusion Detection

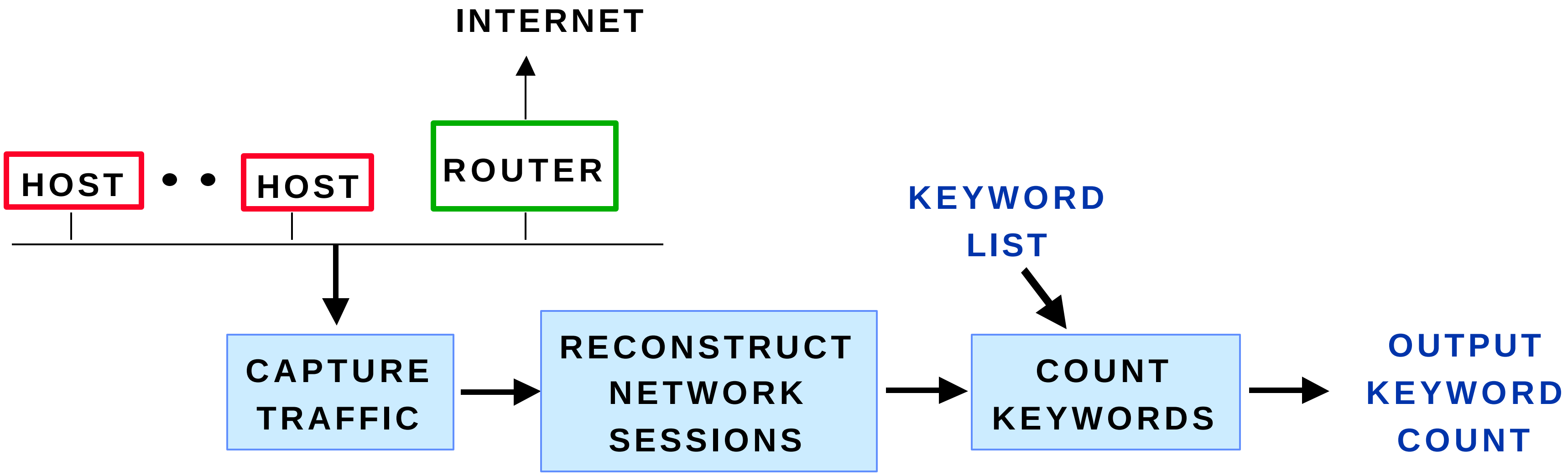


- Network Sniffing Data (NSM, ASIM,EMERALD, BRO, IBM-HAXOR Cisco-NetRanger, ISS-RealSecure, Network Radar, Network Flight Recorder)
- Host Audit Data (STAT, EMERALD, AXENT-Intruder Alert, Centrax)
- Router, Firewall, … Audit Data (Ji Nao, Cisco-NetRanger, EMERALD)

Visibility of Attacks with Different Inputs

	Host Audit	Network Sniffer	Router/ Firewall
Infrastructure			✔
Host Denial of Service		✔	✔
Probes/Scans		✔	✔
Remote to Local	✔	✔	✓
User to Root	✔	✓	✓
Malicious-Code/Data	✔	✓	
Insider Attack	✔		

Sniffer-Based Intrusion Detection



- Popular, Low Cost, No Impact on Hosts, Monitor Many Hosts Simultaneously (ASIM, Cisco-NetRanger, BRO, …)
- Capture Network Traffic, Reconstruct Network Sessions, Count Number of Keywords in Each Session
- Examples of Keyword Strings
  - ftp: root, anonymous
  - login: guest, root, incorrect, daemon, passwd, permission denied

Example Telnet Session Reconstruction

```
HP-UX hqdadev A.09.03 D 9000/750 (ttyt1)
login: ~tftp
Password:
Login incorrect
login: efs
Password
Login incorrect
login: efs
Password:
Password:
/usr/efs w

3:03pm  up 14 days, 21:33,  1 user,  load average: 0.00, 0.00
User     tty           login@  idle   JCPU   PCPU  what
efs      pty/ttyt1     3:03pm     1                w
/usr/efs> cd /

ls -al

total 16336
drwxr-xr-x  47 root     sys         3072 Sep 23 10:41 .
drwxr-xr-x  47 root     sys         3072 Sep 23 10:41 ..
drwx------   2 root     mail        1024 Nov 28  1994 .elm
-rw-------   1 bin      bin         8690 Jul  9 15:39 .profile
-rw-------   1 root     sys           37 Nov 18  1994 .rhosts
dr-x------   3 root     other       1024 May 26  1994 .secure
drwxr-x---   3 root     sys         1024 Jul  3 14:30 Perl
grep :0: /etc/passwd

root:*:0:3:Beginning of All Things...,,976-HPUX,:/:/bin/ksh
```

Problems With Keyword-Based Systems

- High False Alarm Rates
  - Little Use of Context Around Keywords
  - Humans Select Keywords to Detect Attacks With Little Thought of Impact on False Alarms and Little Validation
  - Many Systems Produce 100’s of False Alarms Per Day
  - Keywords Accumulate for Old Attacks and May Generate False Alarms for New Types of Normal Network Traffic
  - Requires Knowledge Of Attack Details, Sometimes Difficult To Select Keywords to Detect an Attack
- Misses New Attacks, May Miss Attack Variants, Requires Constant Updating (Like Virus Detection)
  - Keywords are Often Too Attack Specific and Depend on Visibility of Attack Script and Use of an Unchanging Script
- Does not Provide a Correct Attack Name
  - It is Often Difficult to Infer the Attack Name from Keyword Counts

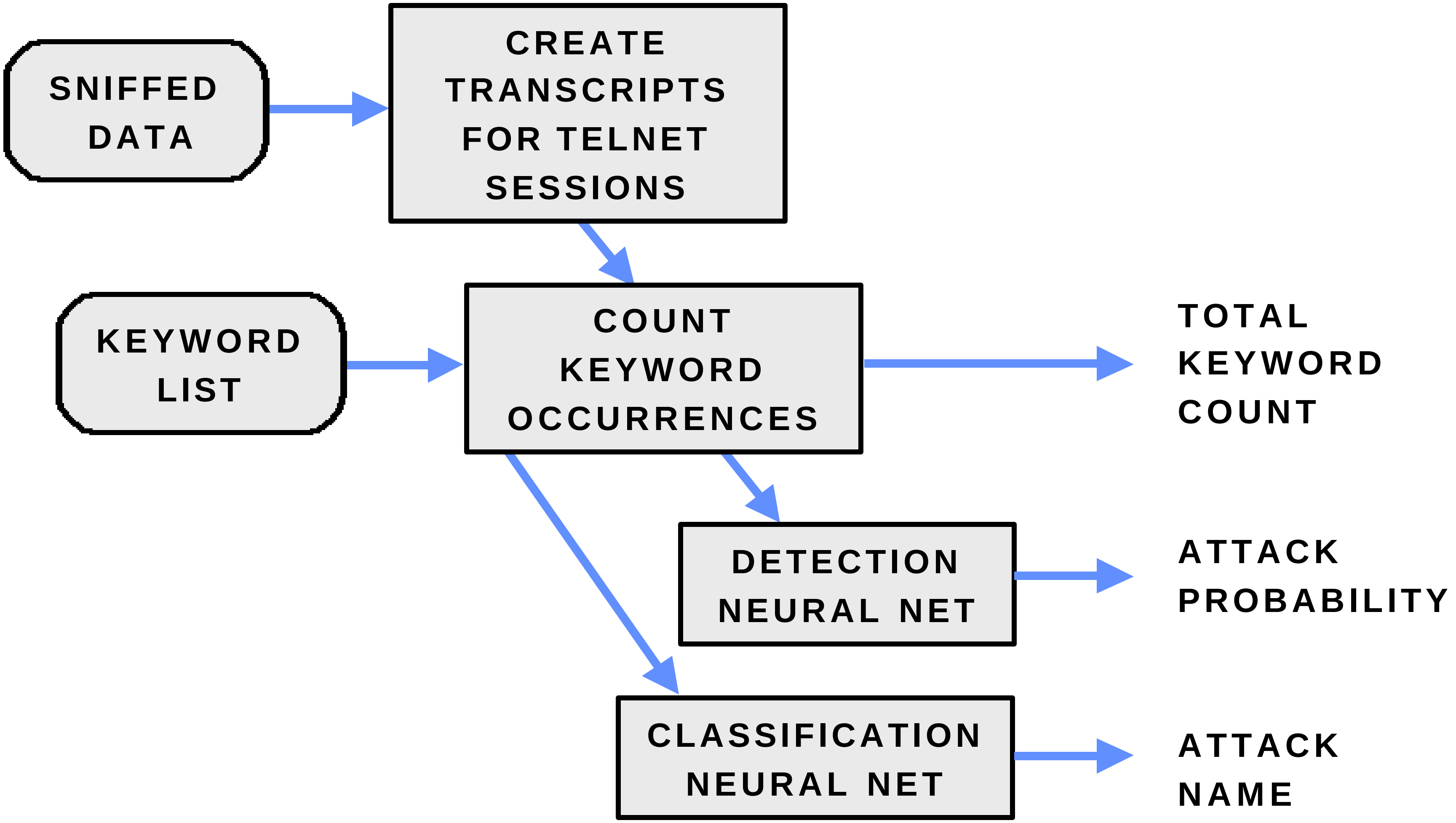
Outline

- Intrusion Detection Background
- → Goals and System Design
- Evaluation and Results
- Summary
- Future Work

Goal of This Work

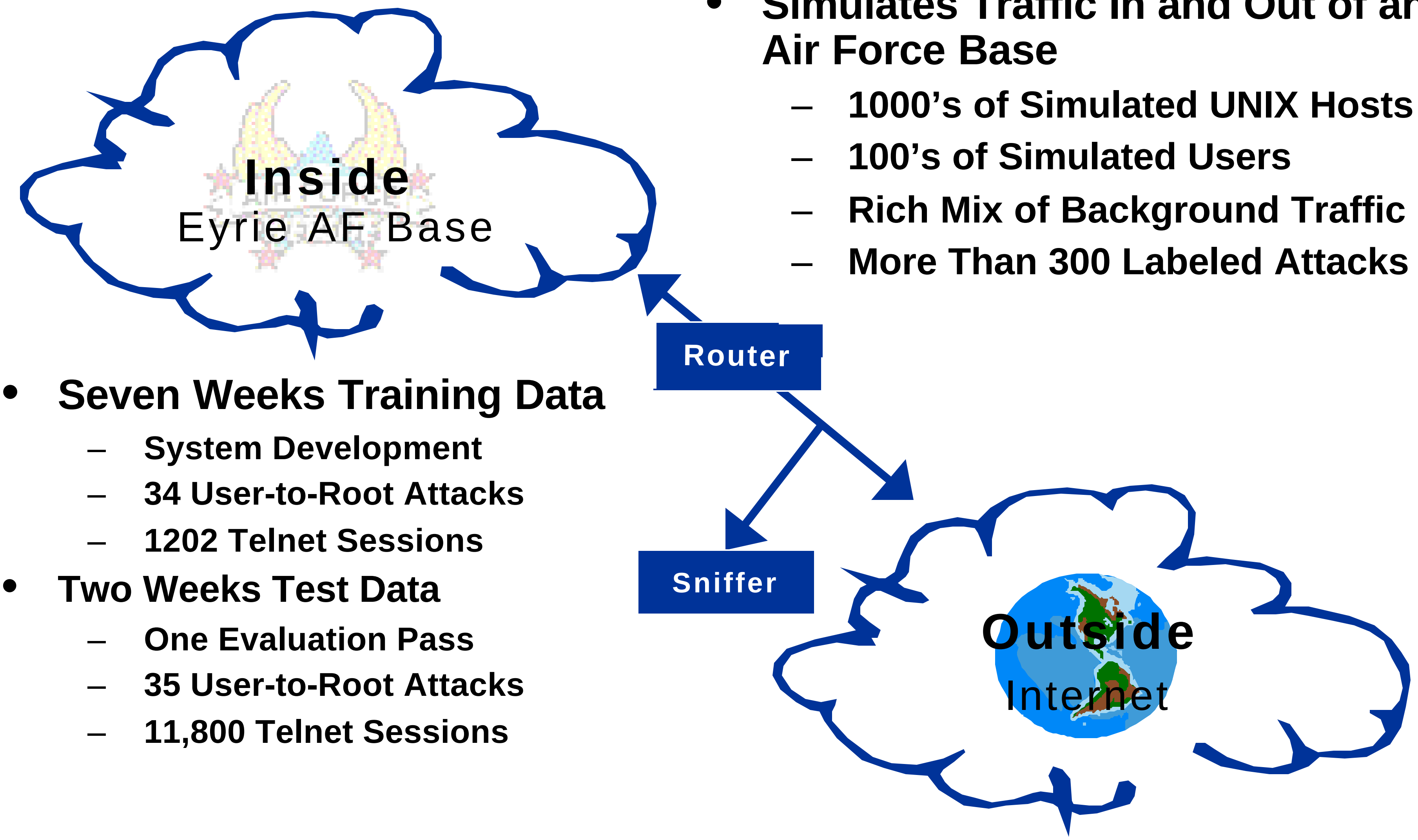
- Improve Performance Of Existing Keyword-Based Systems
  - Use Neural Networks and Automatic Training on Normal Data and Attacks to Select Keywords and Keyword Weightings that Provide Good Detection and Few False Alarms
  - Select More Robust Keywords that Can Detect New and Old Attacks
- Focus on UNIX Attacks Where Users Illegally Become Root
  - This is a Difficult, but Important Class of Attacks
- Determine if Neural Networks can Provide Attack Labels
- Constraints
  - To Permit Retrofitting in Existing Systems, Continue to Use Keyword Counts in Telnet, Rlogin, and other Sessions
  - Use Neural Network to Postprocess Keyword Counts

Approach for Attack Detection and Classification



Training and Test Data from DARPA 1998 Intrusion Detection Evaluation

- Simulates Traffic In and Out of an Air Force Base
  - 1000’s of Simulated UNIX Hosts
  - 100’s of Simulated Users
  - Rich Mix of Background Traffic
  - More Than 300 Labeled Attacks



- Seven Weeks Training Data
  - System Development
  - 34 User-to-Root Attacks
  - 1202 Telnet Sessions
- Two Weeks Test Data
  - One Evaluation Pass
  - 35 User-to-Root Attacks
  - 11,800 Telnet Sessions

Eject Attack Example from Training Data

1. Gain User Access:
Attacker Logs Into Telnet
Using Sniffed Password

2. Download Attack Code:
Type Directly into
uudecode to Hide Keywords

3. Preparations:
Compile Attack Programs
Using gcc With
Innocuous Names

4. Run Attack:
Buffer Overflow Creates
Root Shell

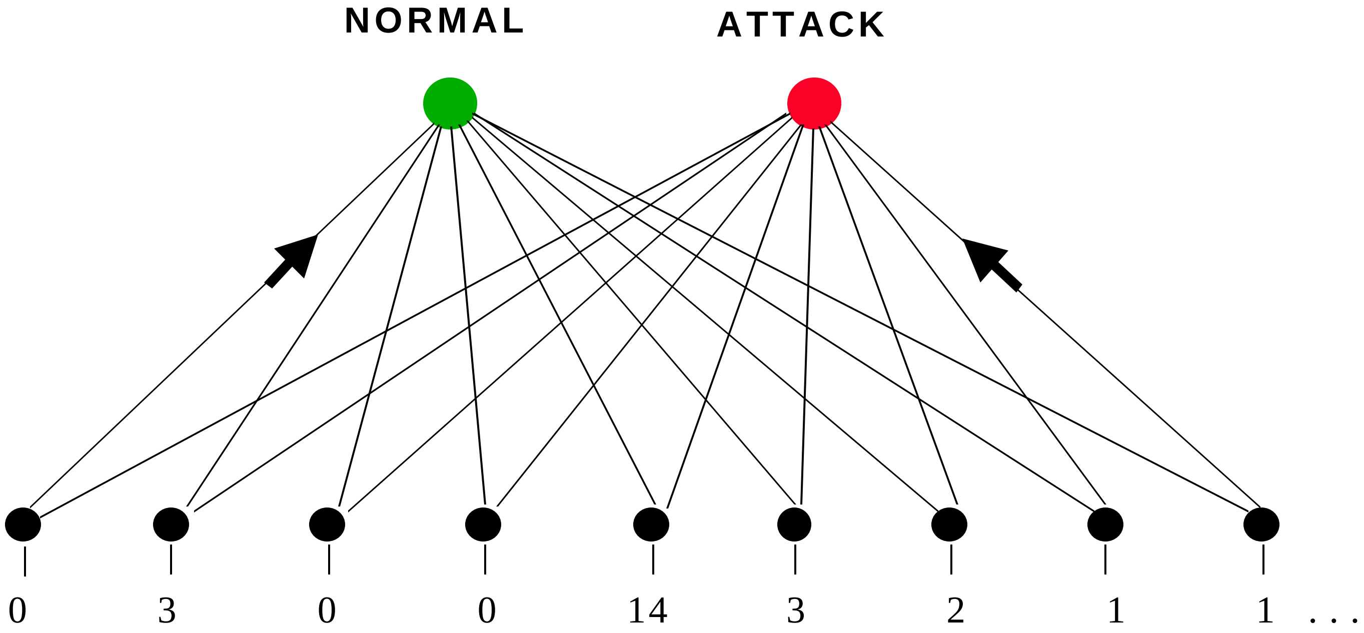
5. Actions:
Exit Root Shell and
Logout, Attack Verified

```
UNIX(r) System V Release 4.0 (pascal)
@#$[CR]$#@@#$[0]$#@
@#$[CR]$#@@#$[0]$#@login: alie
Password:
Last login: Wed Jul  1 16:12:34 from 194.27.251.21
Sun Microsystems Inc.   SunOS 5.5       Generic November 1995

Official U.S. government system for authorized use only. Do not discuss,
enter, transfer, process or transmit classified/sensitive national security
...

pascal> @#$[CR]$#@@#$[0]$#@
pascal> which gcc@#$[CR]$#@@#$[0]$#@
/bin/gcc
pascal> uudecode<<XX899347368XX\`@#$[CR]$#@@#$[0]$#@
? begin 644 /tmp/17857.c@#$[CR]$#@@#$[0]$#@
? M(VEN8VQU9&4@/'-T9&EO+F@^"B-I;F-L=61E(#QS=&1L:6(N:#X*(VEN8VQU@#$[CR]$#@@#$[0]$#@
...
? M(&)U9ELQ72P@*&-H87(@*BD@,"D["B`@<&5R<F]R*")E>&5C;"!F86EL960B@#$[CR]$#@@#$[0]$#@
? `@#$[CR]$#@@#$[0]$#@
? end@#$[CR]$#@@#$[0]$#@
? XX899347368XX\`@#$[CR]$#@@#$[0]$#@
pascal> /bin/gcc -o /tmp/178572 /tmp/17857.c@#$[CR]$#@@#$[0]$#@
pascal> which gcc@#$[CR]$#@@#$[0]$#@
/bin/gcc
pascal> uudecode<<XX899347375XX\`@#$[CR]$#@@#$[0]$#@
? begin 644 /tmp/17857.c@#$[CR]$#@@#$[0]$#@
? M(VEN8VQU9&4@/'5N:7-T9"YH/@IV;VED"FUA:6XH:6YT(&%R9V,L(&-H87(@@#$[CR]$#@@#$[0]$#@
? M*F%R9W9;72D*>PH@('-E=')E=6ED*#`L,"D["B`@97AE8VPH(B]B:6XO=&-S@#$[CR]$#@@#$[0]$#@
? /:"(L(G1C<V@B+#`I.PI]@#$[CR]$#@@#$[0]$#@
? `@#$[CR]$#@@#$[0]$#@
? end@#$[CR]$#@@#$[0]$#@
? XX899347375XX\`@#$[CR]$#@@#$[0]$#@
pascal> /bin/gcc -o /tmp/178573 /tmp/17857.c@#$[CR]$#@@#$[0]$#@
pascal> /tmp/178572@#$[CR]$#@@#$[0]$#@
Jumping to address 0xeffff7e0
Jumping to address 0xeffff7e0 B[364] E[400] SO[400]
#
/tmp/178573
# # #
^D@#$[BS]$#@@#$[BS]$#@# @#$[CR]$#@@#$[0]$#@
# exit
#
^D@#$[BS]$#@@#$[BS]$#@#
pascal> @#$[CR]$#@@#$[0]$#@
pascal> ^D@#$[BS]$#@@#$[BS]$#@logout
```

Keywords for User to Root Attacks

- Initially We Had 58 Old Keywords Commonly Used in Existing Intrusion Detection Systems
  - Detect Suspicious Actions (“passwd”, "+ +”, “daemon”, "warez ”, “shadow”, “permission denied”, “showmount”)
  - Detect Old Attacks ("from: \|”, "CWD ~ROOT”, "LD_PRELOAD”, "login: guest ”)
- Added 31 New Keywords Based on Training Data
  - Detect Root shell (“root:”, “uid=0(root)”)
  - Detect Setup Actions (“chmod”, “gcc”)
  - Detect Attack Code Downloading (“uudecode”, “<<“, “>ftp get”)
  - Attack Specific (“IFS=“, “FDFORMAT”, “FFBCONFIG”)
  - Detect Operating System("SunOS UNIX”, "Red Hat Linux”)

Simple Single-Layer Network Provided Good Performance on Training Data



KEYWORD COUNT INPUTS

- Inputs Are Counts of the Number of Key Words in a Telnet Session
- The Two Outputs Estimate Posterior Probabilities for Normal Sessions and Attacks (Squared-Error Stochastic Gradient Descent)
- Feature(Keyword)-Selection and Training/Testing Performed Using 10-Fold Cross-Validation

Neural Net Detection Keywords

- 10-Fold Cross-Validation Testing on the Training Data Demonstrated that Best Detection Performance Was Obtained with Only 30 Keywords
  - Fewer Keywords Decreased Detection Rate
  - More Keywords Increased False Alarm Rate
  - Fewest Cross-Validation Errors (1 Miss, 1 False Alarm) at 30 Keywords
- Top Ten Keywords (Specified Using Perl Regular Expressions)

```
"cat\s*>"
"Jumping to address"
"begin [0-9]"
"uudecode\s*[\<\-]"
"linsniff"
"uid\=0\(root\)"
"\$\>\=0\;\$\<=0\;"
"login\: guest\s"
"ffbconfig"
"^bash\#\s"
```

User-to-Root Attack Types in Test Data

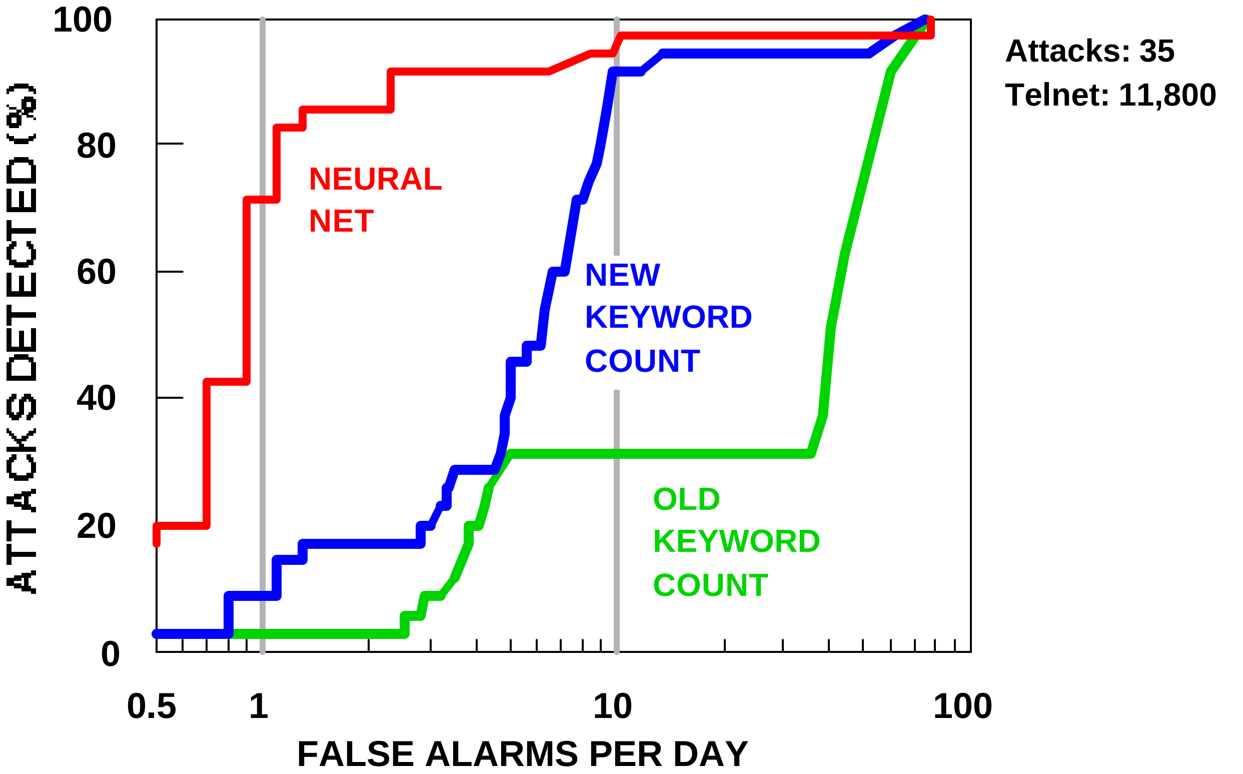
	Solaris	SunOS	Linux
OLD	eject ffbconfig fdformat	loadmodule	perl
NEW	ps	ps	xterm

- Seven User-to-Root Attack Types, 35 Instances in 11,859 Telnet Sessions
- Different Techniques Used to Encrypt, Transport, Prepare Script, Different Actions After Breakin, Some Attacks Spread Over Multiple Sessions
- Although This Test was not Part of the Official DARPA Evaluation, No Part of Test Data was Used During System Design or Training and the Evaluation Rules Were Followed for Testing

Outline

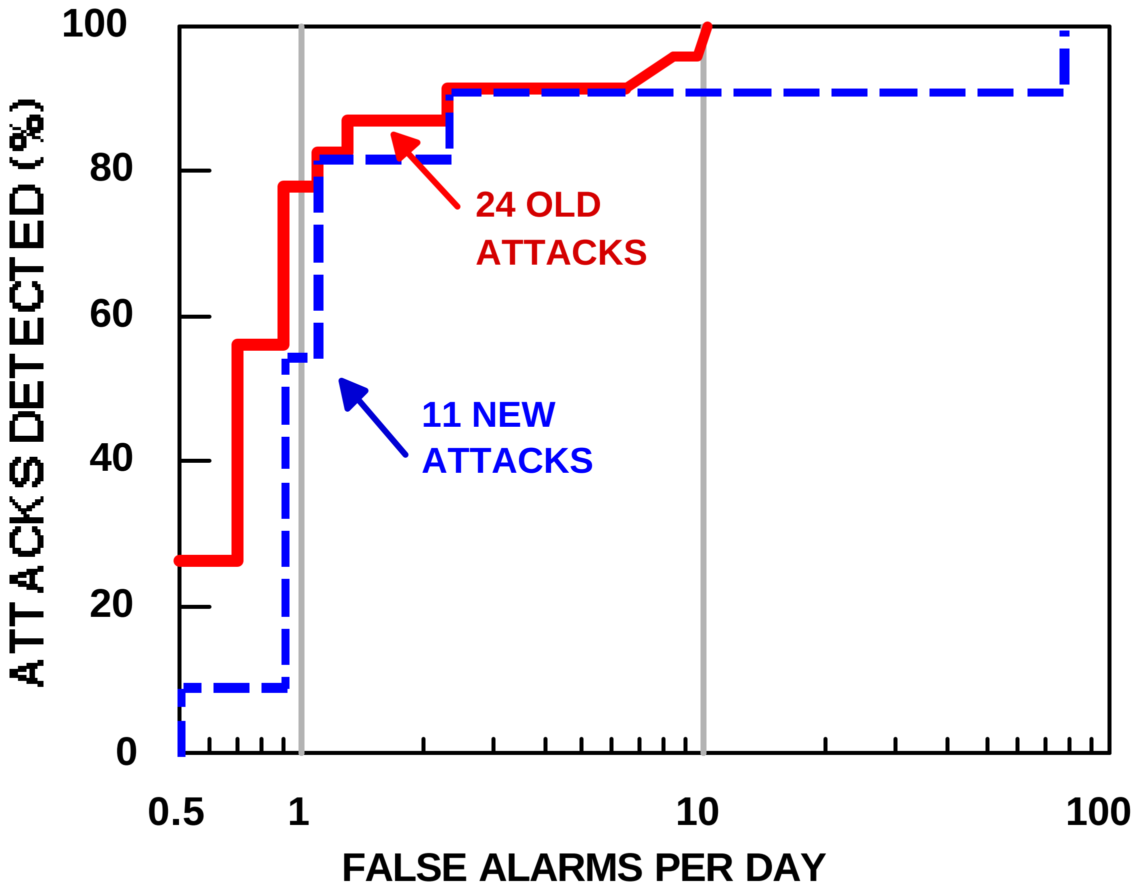
- Intrusion Detection Background
- Goals and System Design
- → Evaluation and Results
- Summary
- Future Work

Overall Test Results,
User to Root (u2r) ROCs



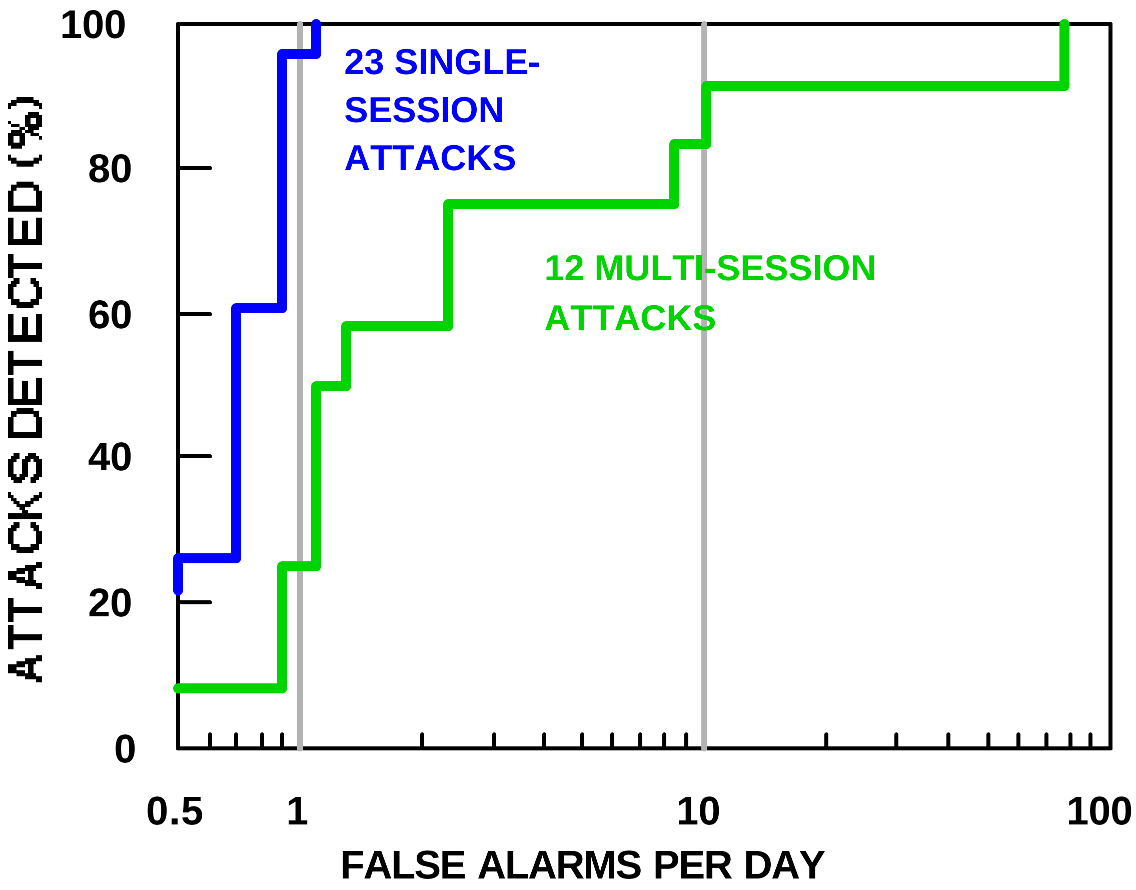
- Trained Neural Net With New Keywords Provides Best Performance (Roughly 80% Detection at 1 False Alarm Per Day)
- Keyword Count with Old Keywords Provides Poor Performance (100 False Alarms Per Day for Good Detection)
- Adding and Selecting 30 Keywords Reduces False Alarm Rate by x10

Neural Network Detection of Old versus New Attacks



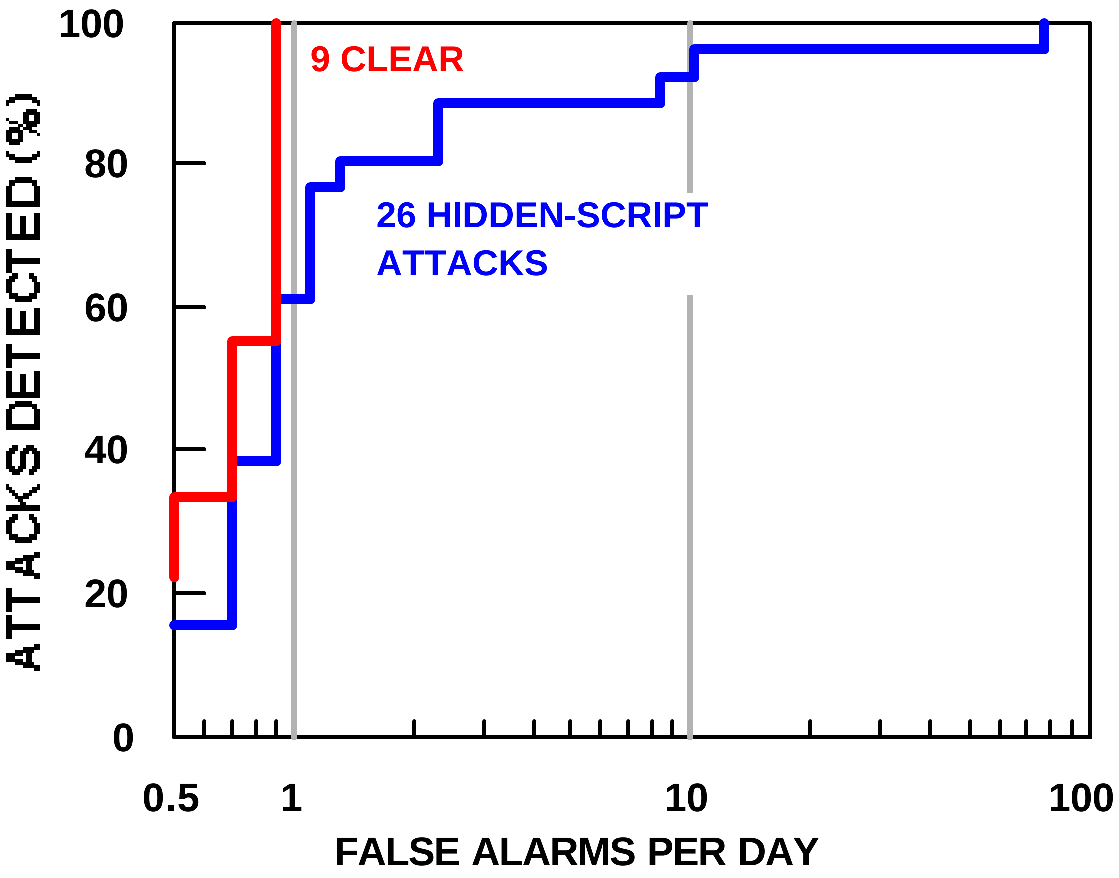
- Good Detection of Both Old and New Attacks Due to Common Script Transport and Preparation Mechanisms and Actions

Stealthy, Multi-Session Attacks Versus Single Session Attacks



- **Multi-Session Attacks More Difficult to Detect**
  - **Setup (Exploit Script Transmission) and Breakin Occur in Separate Sessions and Clues are Dispersed Over Time**

Clear-Text Exploit Transmission Versus Hidden Transmission



- Attacks Where the Exploit is Visible as Clear Text are Easier to Detect than Attacks With Hidden Text

Attack Classification for Clear-Text Attacks

Desired	Computed Class				
	Solaris			SunOS	Linux
Class	eject	format	ffb	loadm	perl
-----					
eject	2				
format		1			
ffbconfig			1		
loadmodule					
perl					1

- **Perfect Classification for Clear-Text Attacks 100% Correct**

Desired	Computed Class				
	Solaris			SunOS	Linux
Class	eject	format	ffb	loadm	perl
-----					
eject	6				
format	5	1			
ffbconfig	1		1		
loadmodule				2	
perlmagic					2

- **Within-Operating-System Errors for Encrypted Attacks 67% Correct**

Outline

- Intrusion Detection Background
- Goals and System Design
- Evaluation and Results
- Summary
- Future Work

Summary

- Using A Neural Network with Extended Keyword Strings Provides a High-Performance Intrusion Detection System for the DARPA 1998 Test Data (Unofficial Results)
  - Dramatically Lower False Alarm Rate
  - False Alarm Rate Near 1 per Day, Detection Rate > 80%
  - Finds Both Old and New Attacks
  - Detects Many Attack Components(e.g. setup, breakin, actions)
  - Training Provides Automatic Keyword Selection and Weighting to Minimize the False Alarm Rate and Maximize Detection

Future Work

- Embed Neural Network Approach in Existing System
  - Add New Keywords
  - Use Detection Neural Network to Compute Score
  - Use Identification Neural Network to Label Attacks
- Potential Improvements
  - Use Recent Attacks and Traffic to Improve Keywords and Scoring
  - Integrate Information Across Multiple Telnet Sessions and Services (e.g. ftp).
  - Add strings to detect additional approaches to download code and prepare for an attack (e.g. vi, mail, ..) and additional actions.
  - Make use of Context around Strings.