## Backscatter and Global Analysis
Stefan Zota

- **Analyze tcpdump logs and get global abnormal traffic**
- **Gather vanilla statistics:**
  - **Port classification traffic**
  - **Destination and Source address traffic**
  - **Classification of the ICMP error messages (TTL field, port unreachable, fragmentation needed, echo reply for echo request floods)**
  - **Percent of successful connections and connection based bit rate and packet rate stats**
  - **Average duration and interval for malicious connections**



## Attacks

- **Top attack sources**
- **Unsuccessful TCP connections**
  - **SYN attacks**
  - **TCP RST ACK backscatter**
- **DNS lookup and ping messages to check if we have spoofed inexistent sources**
- **Look at the destination IP over fixed time-windows. Event based classification**



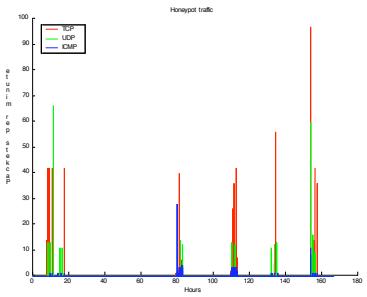
## Intrusion Detection with Honeypots
Claire O'Shea

- Setup
  - Using honeyd to monitor 14 IP addresses in the CS department
  - Behind campus IDS, but not department firewalls
  - All incoming packets are logged
  - 1 week of passive monitoring
  - 1 week of emulating services (in progress)
    - FTP server
    - Telnet
    - SMTP server
    - Web server

## Intrusion Detection with Honeypots

Passive monitoring results:

- 1491 packets received from 60 unique IPs
  - ICMP: 214 packets from 27 IPs
  - TCP: 951 packets from 33 IPs
  - UDP: 326 packets from 3 IPs




## Intrusion Detection with Honeypots

Active monitoring results:

- Comparable amount of traffic (lots of scans)
- So far, most of interesting activity has been on port 80

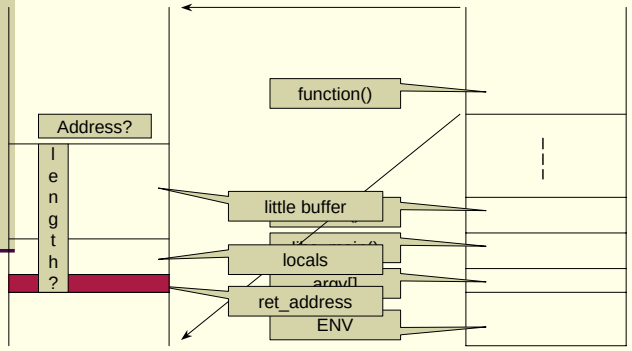
```
Sun Apr 24 10:41:52 EDT 2005: Connection from
61.73.157.174 to 152.2.137.202 Port 80
POST /_vti_bin/_vti_aut/fp30reg.dll HTTP/1.1
Host: monaco137.cs.unc.edu
Transfer-Encoding: chunked
Content-Length: 1499

(lots of binary data)
```

An IIS server script with a known buffer overflow exploit!

## About that little buffer in that Big Server

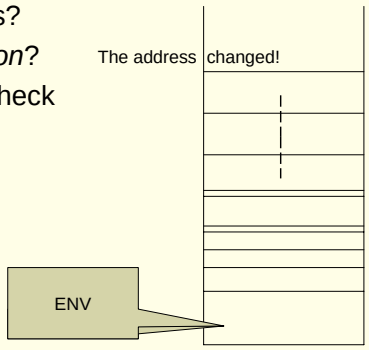
Ritesh Kumar

## The Exploit




## The Counter

- Forking Servers?
- *Stack Translation*?
- Array bounds check
  - C#, Java
  - C !!
- libsafe




# Simulation of DoS Attacks

Ben Wilde

## Experiment Description

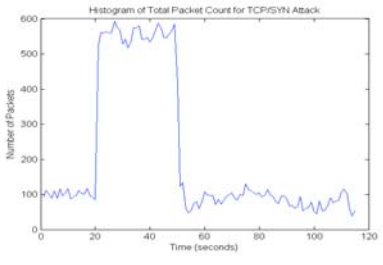
- Setup
  - One 'attacker' and one 'victim'
    - tcpdump on victim, MACE on attacker
- Tools
  - "MACE"
    - Malicious traffic generator from Paul Barford's group at Wisconsin
    - Pre-programmed series of attacks*
      - TCP/SYN
      - Rose
      - Teardrop
      - Ping of Death
      - Bonk
      - Jolt
      - Land
      - Nestea
      - Oshare

*I still don't know what most of them do or if they are interesting…

## Preliminary Results

I finally got everything running on Tuesday and ran each of the attacks listed on the previous page. I still need to look into what the attacks are and then crunch the data to see what looks interesting…

Example histogram from a TCP/SYN attack:




## *Using LDA classification to Identify Malicious traffic*

Alok Shriram

## Objective

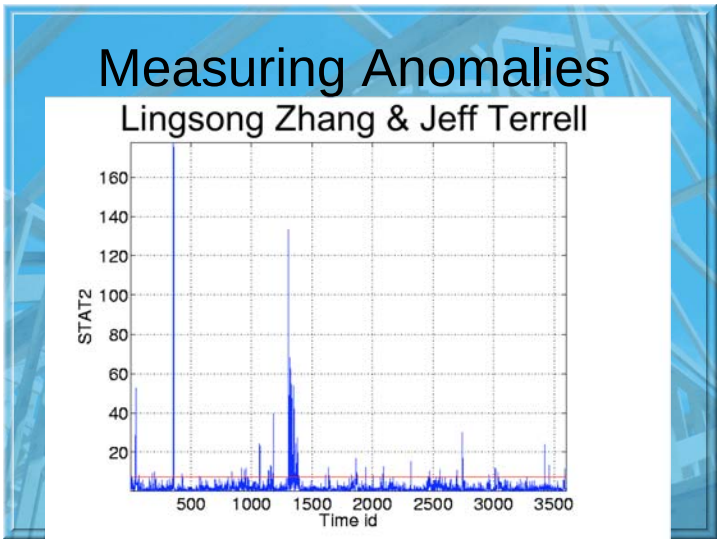
- Previously had talked about classification techniques.
- LDA proposed a classification scheme to classify traffic for different QoS
- **Project Goal: To use the LDA to see if we can distinguish attack and normal traffic**

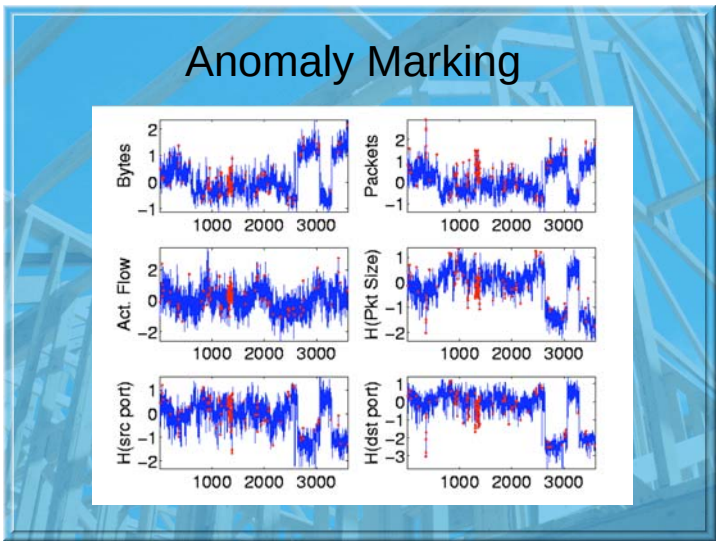
## Recap

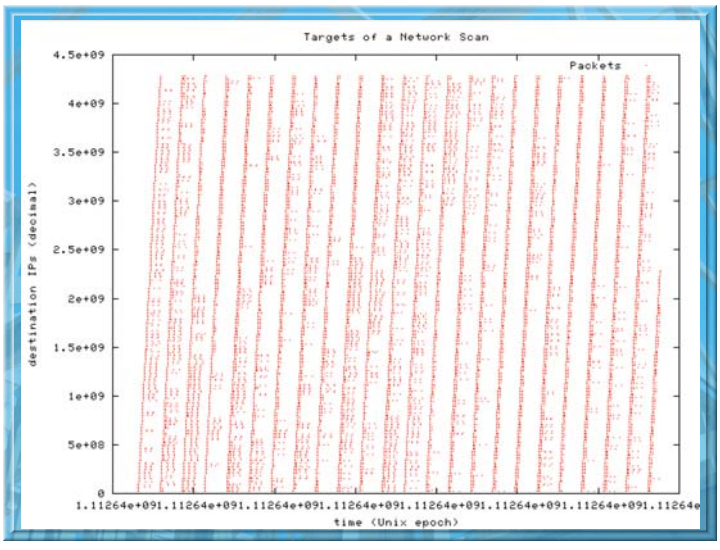
- LDA uses some training data points with known classification
- With those data points it attempts to classify new data points
- Each of the training data points needs to be on a metric which characterizes that connection

## Metrics in Consideration

- Average Packet size
- Root Mean Square Packet size
- Average Flow duration
- Average IAT
- Variance of IAT
- Number of packets
- Number of packets per second



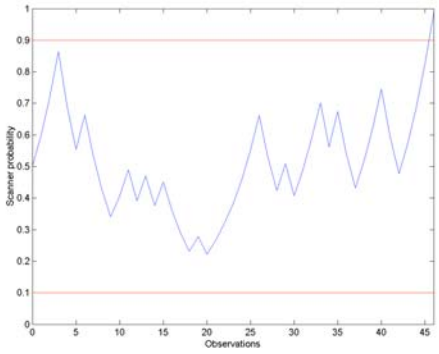



# CloudShield Portscan Detector

Brian Begnoche

- Threshold Random Walk algorithm
  - "Fast Portscan Detection Using Sequential Hypothesis Testing"
  - Implement on CloudShield
  - So far, TCP only and unidirectional
- Observe connection attempts from source
  - Failure indicative of scanner
  - Look for SYN of 3-way handshake from source, if ACK observed later then it is a successful attempt
  - After enough observations, classify as scanner or benign
- Still working out the kinks...
- Will compare with ATN scanner alarms

# Example TRW




# Effectiveness of Shrew Attack on Real Networks

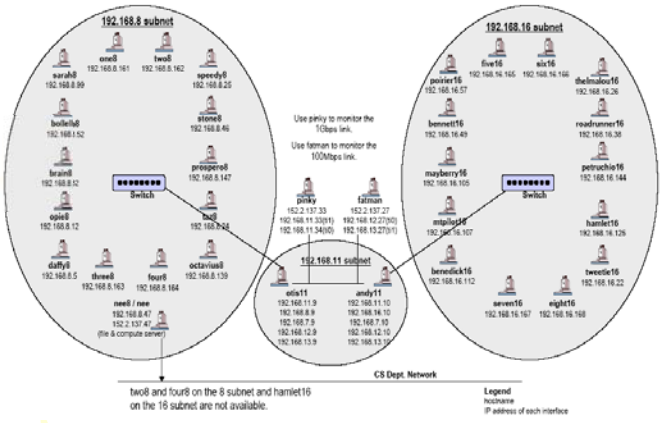
**By**

**Sushant Rewaskar**

# Tmix trace generator

- Generating realistic traffic is difficult due to TCP feedback loop
- Source level models like SURGE are difficult to understand, model and more so to maintain
- Tmix uses an application and network independent model
  - It uses Application data units instead of network data units.

# Experimental network

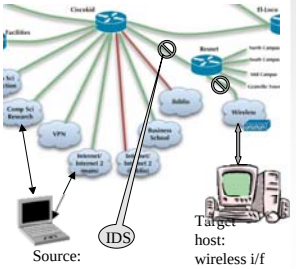



# Evading UNC's IPS/IDS

Priyank Porwal

Trying out Evasion, Insertion techniques to bypass Tipping Point and Snort protecting UNC network

- Goal: Test instead of Attack
- Tests similar to those done by Ptacek and Newsham
  - Different TTL values
  - DF bit set and large packets
  - IP fragment, TCP segment reassembly related issues
- Dummy attack string instead of strings detected by IDS
  - To avoid being blocked totally



## Current Status

- Able to establish TCP connections by sending raw packets and sniffing responses
  - Firewalled some ports on my laptop used to attack
- Developed a shell to take different parameters: TTL, Seq#, Ack#, Data, etc. before hand-crafting packets and sending them out

```
airborne:~/Projects/nids porwal$ sudo ./run en1 152.23.73.50 6969 152.23.75.10 2
5501
Password:
Command > send d=credit
152.23.73.50:6969 -----> 152.23.75.10:25501
Seq=651159046    Ack=1804289422  Flags=_A____   Length=0
Data=
Command > send d=bank
152.23.73.50:6969 -----> 152.23.75.10:25501
Seq=651159046    Ack=1804289426  Flags=_A____   Length=0
Data=
Command > send s=1804289422 d=hola
152.23.73.50:6969 -----> 152.23.75.10:25501
Seq=651159046    Ack=1804289426  Flags=_A____   Length=0
Data=
```

## Results / Still To Do

- Findings
  - TCP connections do not timeout on Windows XP, even after almost 60 hrs of inactivity
- Tests
  - Framework and infrastructure ready
  - Tests still to be run

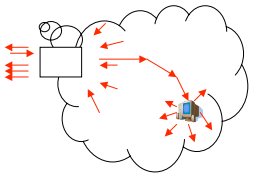
# Header Based 'Attack Control' Schemes for Worm and DDOS Threats

Comp290: Network Intrusion Detection

Manoj Ampalam & Ankur Agiwal

## Worm Attack Control :

- Utilize packet propagation properties
  - WORM:
    - Tries to multiply upon reaching a compromised machine.
    - For one worm packet coming in, multiple packets targeting multiple destinations go out.
    - Target a particular vulnerability – fixed port.
    - For a particular port, study number of distinct destination address reached out

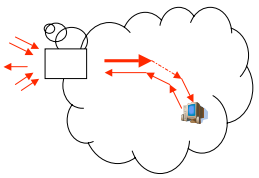


## Worm Attack Control :

- Algorithm:
  - *for a time interval **T***
    *for a port number*
    *if there are some incoming packets*
    *if(#(distinct out_destination address) > (1 + **delta**)average))*
    *suspect attack;*
    *endif*
    *average = **alpha**(average) + (1-**alpha**)#*
    *end if*
    *end for*
    *end for*

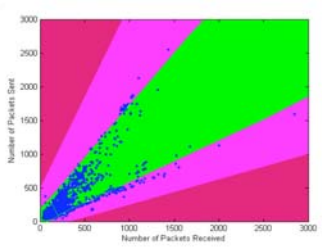
## :DDOS Attack Control :

- Considered Scenario:
  - All internal clients trusted
  - Multiple zombies attack from outer network
  - Network link or CPU resources in victim become limited
  - Change in relationship between
  - Number of incoming and outgoing packets



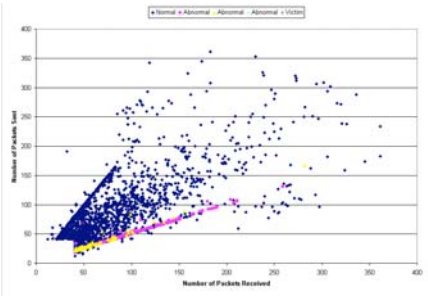
## :DDOS Attack Control

- Using Properties of Normal Traffic Modes:
  - TCP: flow controlled by continuous flow of acknowledgements




## :DDOS Attack Control

- On one of attack data sets (MIT Lincoln Laboratory)



## :DDOS Attack Control

- DDOS:
  - Using Properties of Normal Traffic Modes:
    - ICMP: Behavior Similar to TCP
      - Each request associated with corresponding reply
    - UDP:
      - Limit Number of allowed connections
      - Limit Number of packets per connection

# Finding Representative Records from tcpdump dataset

Feng Pan

## Data Set

- Summarize tcpdump raw data into connection records.

| Time stamp | duration | service | Src_host | Dst_host | Src_byte | Dst_byte | flag |
|---|---|---|---|---|---|---|---|

Summarize connection according to the SYN and FIN flag?
How to get the service type?
Where is the flag such as “S0, SF, REJ” in the raw data?

- Discretize numeric attributes, and make all attributes categorical.

## Find representative records

- The dataset is unlabelled, therefore, unsupervised data mining method shall be used.
- Algorithm: Representative Finder
  - Similar to clustering algorithm
  - Selects representatives which cover both major types (normal connections) and outliers (intrusion connections).
  - Works well on biased dataset. (intrusion connections shall be a small portion in the dataset)

## Expected Results

- Find representative records which can cover both normal connections and abnormal connections.
- New connection records can be clustered according to the representative records which work as cluster centers.

## Flash Crowds and DoS Attacks

Eric Stone

*The* UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

## Characteristics

- Both flash crowds and DoS attacks manifest themselves similarly
- In both cases the end system sees a sudden spike in activity that may result in a reduced availability of resources



## Characteristics

- However, by definition a DoS attacks is malicious and undesirable while a flash crowd is an increase in desired activity
- An end system wants to block a DoS attack and in most cases encourage increases in usage
- The end system may wish that usage increases be more gradual than a flash crowd



## Characteristics

- End system have options for defending themselves against DoS attacks
- They don't want to use these defenses against flash crowds as that would discourage potential legitimate users
- They need to have a way to distinguish between the two



## Project

- My project will involve looking at the data traces available to class
- These traces are being checked for significant spikes in activity
- These spikes are likely to represent either a flash crowd or a DoS attack against an end system

## Project

- These located spikes will then be checked for defining characteristics that could help identify them as either a flash crowd or a DoS attack
- The defining elements include such factors as the slope of the attack edge, the behavior of the attack sources on the end system, the distribution and previous contacts with the attack sources



## Project

- While both flash crowds and DoS attacks involve sudden spikes in usage, there tends to be sharper spike for DoS attacks
- A flash crowd will behave more interactively and naturally with the end system
  - This will be harder to determine given the nature of the traces



## Project

- Flash crowds tend to have a more natural source distribution than DoS attacks do
- Flash crowds also tend to include more previously seen hosts



## Project

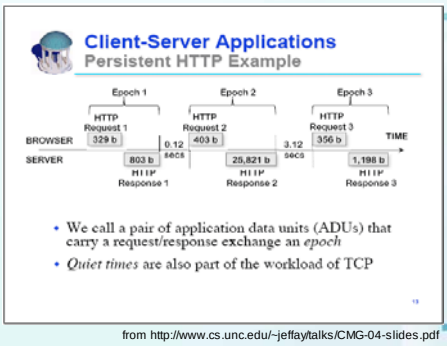
- Once the spikes have been classified as flash crowd or attacks, I will attempt to determine out of band whether or not they really were attacks
- This has potential uses in attack mitigation but runs the risk of becoming an arms race between attacker and defender



# a-b-t Model & CloudShield

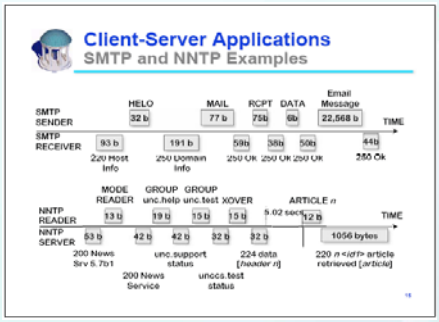
Boriana Ditcheva
Lisa Fowler
Elise London
Dr. Kevin Jeffay & Dr. Diane Pozefsky

## The a-b-t Model




from http://www.cs.unc.edu/~jeffay/talks/CMG-04-slides.pdf

## The a-b-t Model




from http://www.cs.unc.edu/~jeffay/talks/CMG-04-slides.pdf

## So what can we do with this?

- **Using this model, with no additional semantic knowledge, we can determine:**
  - **what services are being used**
  - **what's normal**
  - **if something is anomalous**

## Why CloudShield?

- **Previously, analysis was done offline**
- **With this powerful inline machine, we can:**
  - **collect the epoch data quickly**
  - **report it**
  - **act on it!**