Evading/Attacking NIDS

Priyank Porwal

COMP 290

Network Intrusion Detection Systems

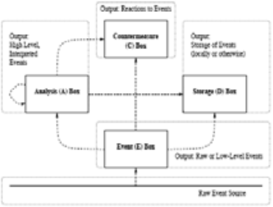
Agenda

- **Network ID Systems**
  - **Architecture, Problems**
  - **Insertion, Evasion, DoS Attacks**
- Proposed Solutions
  - Traffic Normalization
  - Active Mapping
- Miscellaneous
  - Evasion with Unicode
  - Evasion using Polymorphic Shell Code

NIDS Architecture

Sets of **Common Intrusion Detection Framework** (CIDF) components

- E-boxes (Event generators)
  - E.g. Sniffers, Monitors
- A-boxes (Analysis engines)
  - E.g. Signature matchers
- D-boxes (Storage systems)
  - E.g. Loggers
- C-boxes (Countermeasures
  - E.g. Alarms, Firewalls


Output: Reactions to Events
Countermeasure (C) Box
Output: High Level, Interpreted Events
Analysis (A) Box
Output: Storage of Events (locally or otherwise)
Storage (D) Box
Event (E) Box
Output: Raw or Low-Level Events
Raw Event Source

NIDS Design Considerations

- Logical Target of Attacks
  - Each component a potential point of vulnerability and hence attacks
- Possible Attacks on their
  - “Availability” (total shutdown)
  - “Accuracy” (false positives)
  - “Completeness” (false negatives)
- Need to be Reliable, Robust
  - Avoid false sense of security

Problems with NIDS

- Passive Network Monitors
  - Inherently “fail-open”
  - Cease to provide protection when subverted
- Vulnerability to Denial of Service
  - Process all flows to all protected end-systems
  - Being complex systems require lots of resources
  - Resource starvation problem is not easily solvable

Problems for NIDS [contd…]

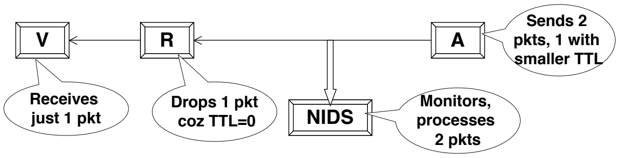
- Insufficient Information on the Wire
  - Not enough to correctly reconstruct the state of complex protocol transactions like at end-systems
- Diversity in Protocol Implementations
  - Packet processing differs across end-systems
  - Leads to ambiguous interpretations
- Unknown Internal Network Conditions
  - Topology, Router configs, Traffic congestion, etc.

Attacks against NIDS

- **Insertion**
  - – Stuffing the analyzer with “invalid” packets
- **Evasion**
  - – Slipping “valid” packets past the analyzer
- **DoS**
  - – Causing resource starvation

Insertion

- NIDS accepts packets that an end-system rejects or doesn’t even receive
  - – Data gets “inserted” into the NIDS’s packet stream



- Occurs when NIDS is ***less strict*** in processing packets than internal network

Insertion Example

- Attacker’s Data Stream

2	3	3	5	4	1	6	Seq#
T	T	X	C	A	A	K	Data

- NIDS’s Stream

Accepts 3rd packet which overwrites 2nd packet data

Interprets “**ATXACK**”

1	2	3	3	4	5	6	Seq#
A	T	T	X	A	C	K	Data

- End-System’s Stream

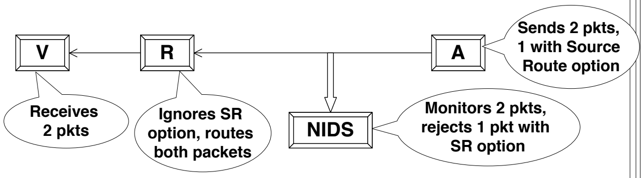
Rejects 3rd packet for some reason, or does not receive it

Interprets “**ATTACK**”

1	2	3	~~3~~	4	5	6	Seq#
A	T	T	~~X~~	A	C	K	Data

Evasion

- An end-system can accept a packet that an NIDS rejects
  - – Data gets “slipped” past the NIDS


V
R
A
Sends 2 pkts, 1 with Source Route option
Receives 2 pkts
Ignores SR option, routes both packets
NIDS
Monitors 2 pkts, rejects 1 pkt with SR option

- Occurs when NIDS is ***more strict*** in processing packets than internal network

Evasion Example

- Attacker’s Data Stream

2	3	3	5	4	1	6	Seq#
T	X	T	C	A	A	K	Data

- NIDS’s Stream

Rejects 3rd packet for some reason

Interprets “**ATXACK**”

1	2	3	~~3~~	4	5	6	Seq#
A	T	X	~~T~~	A	C	K	Data

- End-System’s Stream

Accepts 3rd packet which overwrites 2nd packet

Interprets “**ATTACK**”

1	2	3	3	4	5	6	Seq#
A	T	X	T	A	C	K	Data

Real Insertion/Evasion Attacks

- Mostly leverage on **basic network and protocol ambiguities** at the NIDS
  - – Ambiguous interpretation of header fields
  - – Ambiguous handling of header options
  - – Ambiguous **fragment/segment reassembly**
- Ambiguities can cause NIDS to accept/reject packets differently than the end-system
  - – NIDS and the end-system get different views of the same data stream

Ambiguities at NIDS

Related Field	Ambiguity (Decision problem for NIDS)
TTL	Will the packet reach the end-system before TTL becomes 0?
Length, DF	Will all downstream links be able to transmit this big packet without fragmenting (DF bit set)?
IP Option(s)	Will the end-system/routers accept packet with this IP option(s)? E.g. (Strict) Source Route option
TCP option(s)	Will the end-system accept packet with this TCP option(s)?
Data	Will the end-system accept data in SYN packet?
ToS	Does the packet conform to all internal routers (DiffServ)?
IP Frag Offset	How will the end-system reassemble overlapping fragments?
TCP Seq No.	How will the end-system reassemble overlapping segments?

Reasons for Ambiguities

- Differences in Protocol Implementations
  - – Non-conformance to Protocol Standards
  - – Every OS has a different protocol stack
- Configurations
  - – End-system and router configurations
- Options
  - – Application/Socket level options

IP Fragment Reassembly

- Time-Out
  - – Different fragment time-out periods between NIDS and end-system
  - – Attacker can wait after sending some fragments
    - To let them time-out either at NIDS or at end-system
  - – When should NIDS time-out stored fragments?
    - Storing fragments dropped by end-host (Insertion)
    - Storing fragments for too long (DoS attacks)
    - Dropping fragments stored by end-host (Evasion)

IP Fragment Reassembly [contd…]

- Overlapping Fragments
  - – How will the end-system handle the overlap?
  - – Whether to prefer old or new data?
  - – Different OSs handle overlap differently

Operating System	IP Fragment Overlap Behavior
Windows NT 4.0	Always favors old data
4.4 BSD	Favors new data for forward overlap
Linux	Favors new data for forward overlap
Solaris 2.6	Always favors old data
HP-UX 9.01	Favors new data for forward overlap
Irix 5.3	Favors new data for forward overlap

Transport Layer Ambiguities

- TCP Header Fields
  - – Allow invalid flag combinations?
  - – Accept data in SYN packets?
- TCP Options
  - – Accept/reject options in non-SYN packets?
    - Only if sent and accepted in an earlier SYN
    - MSS (Maximum Segment Size) option in SYN only
  - – PAWS (Protection Against Wrapped Sequence Nos.)
    - End-systems implementing PAWS expect TS (TimeStamp) option in all segments

Transport Layer Ambiguities [contd…]

- TCP 3-way Handshake (TCB creation)
  - – Require full handshake?
    - Misses already active connections
  - – Sync sequence nos. in between?
    - Attacker can easily desync NIDS
    - Best to sync on outbound SYN-ACK packets
- TCP Teardown
  - – When to time-out inactive connections?
    - No implicit TCP connection time-out
  - – FIN and RST to terminate the connection
    - FIN is acknowledged, RST not acknowledged

TCP Stream Reassembly

- Requires Sequence No. Tracking
- Requires Congestion-Window Tracking
  - Normally data past the window is discarded
  - Time lag between NIDS and end-system w.r.t window change events can be a problem
- Missing Data
  - Due to out-of-order arrival or packet drop?
  - NIDS cannot request retransmission

TCP Segment Reassembly [contd…]

- Overlapping Segments
  - How will the end-system handle the overlap?
  - Whether to prefer old or new data?
  - Different OSs handle overlap differently

Operating System	TCP Segment Overlap Behavior
Windows NT 4.0	Always favors old data
FreeBSD 2.2	Favors new data for forward overlap
Linux	Favors new data for forward overlap
Solaris 2.6	Favors new data for forward overlap
HP-UX 9.01	Favors new data for forward overlap
AIX 3.25	Favors new data for forward overlap
Irix 5.3	Favors new data for forward overlap

Denial of Service Attacks

- Basic problem
  - NIDS needs to simulate the operation of all protected end-systems and internal network
- Scarce Resources
  - CPU cycles, memory, disk space, bandwidth
- CPU Cycles
  - Target computationally expensive operations
    - Fragment/Segment reassembly
    - Encryption/Decryption

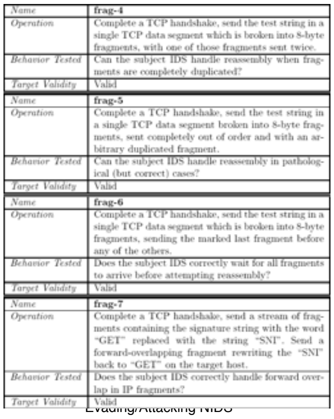
Denial of Service Attacks [contd…]

- Memory
  - Target state management operations
    - TCP 3-way Handshake (TCP Control Block - TCB)
    - Fragment/Segment reassembly
- Network Bandwidth
  - Target NIDS's inability to capture and process packets at line speed
- Reactive Systems
  - Trigger alarms (false positives)
  - Prevent valid access by spoofed addresses

Tests

- Targeted several IP/TCP problems
- Mimicked PHF web-server attack
  - GET /cgi-bin/phf?
  - Possible execution of arbitrary code
  - Supposed to be detected by all NIDSs tested
    - RealSecure
    - NetRanger
    - SessionWalli3
    - Network Flight Recorder (NFR)

Test Examples



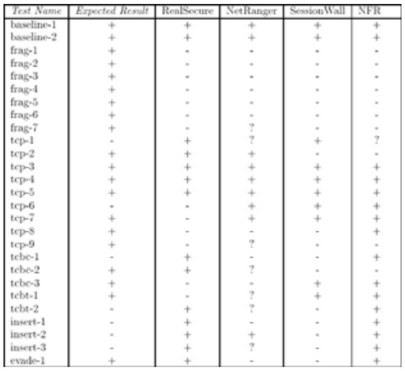
Name	**frag-4**
Operation	Complete a TCP handshake, send the test string in a single TCP data segment which is broken into 8-byte fragments, with one of those fragments sent twice.
Behavior Tested	Can the subject IDS handle reassembly when fragments are completely duplicated?
Target Validity	Valid
Name	**frag-5**
Operation	Complete a TCP handshake, send the test string in a single TCP data segment broken into 8-byte fragments, sent completely out of order and with an arbitrary duplicated fragment.
Behavior Tested	Can the subject IDS handle reassembly in pathological (but correct) cases?
Target Validity	Valid
Name	**frag-6**
Operation	Complete a TCP handshake, send the test string in a single TCP data segment which is broken into 8-byte fragments, sending the marked last fragment before any of the others.
Behavior Tested	Does the subject IDS correctly wait for all fragments to arrive before attempting reassembly?
Target Validity	Valid
Name	**frag-7**
Operation	Complete a TCP handshake, send a stream of fragments containing the signature string with the word "GET" replaced with the string "SNI". Send a forward-overlapping fragment rewriting the "SNI" back to "GET" on the target host.
Behavior Tested	Does the subject IDS correctly handle forward overlap in IP fragments?
Target Validity	Valid

Test Results

+ NIDS detected attack

- NIDS missed attack

? Test could not be conducted



Test Name	*Expected Result*	RealSecure	NetRanger	SessionWall	NFR
baseline-1	+	+	+	+	+
baseline-2	+	+	+	+	+
frag-1	+	-	-	-	-
frag-2	+	-	-	-	-
frag-3	+	-	-	-	-
frag-4	+	-	-	-	-
frag-5	+	-	-	-	-
frag-6	+	-	-	-	-
frag-7	+	-	?	-	-
tcp-1	-	+	?	+	?
tcp-2	+	+	+	-	-
tcp-3	+	+	+	+	+
tcp-4	+	+	+	+	+
tcp-5	+	+	+	+	+
tcp-6	-	-	+	+	+
tcp-7	+	-	+	+	+
tcp-8	+	-	-	-	+
tcp-9	+	-	?	-	-
tcbc-1	-	+	-	-	+
tcbc-2	+	+	?	-	-
tcbc-3	+	-	-	+	+
tcbt-1	+	-	?	+	+
tcbt-2	-	+	?	-	+
insert-1	-	+	-	-	+
insert-2	-	+	+	-	+
insert-3	-	+	?	-	+
evade-1	+	+	-	-	+

Avoiding Insertion/Evasion

- **Problem**
  - **NIDS and end-systems (and internal network) interpret the packets differently**
- How about making NIDS
  - Strict? …. Might cause Evasion attacks
  - Lax? .... Might cause Insertion attacks
- **Ideal Solution**
  - **NIDS interprets the packets exactly like end-systems (and internal network)**

Agenda

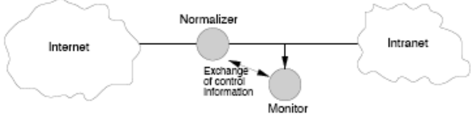
- Network ID Systems
  - Architecture, Problems
  - Insertion, Evasion, DoS Attacks
- **Proposed Solutions**
  - **Traffic Normalization**
  - **Active Mapping**
- Miscellaneous
  - Evasion with Unicode
  - Evasion using Polymorphic Shell Code

Traffic Normalization

- What?
  - Removal of ambiguities from the packet stream before NIDS monitors it
- How?
  - Modifying packet stream to conform to protocol standards
  - Modifying packet stream to conform to internal network topology/configurations

Normalizer

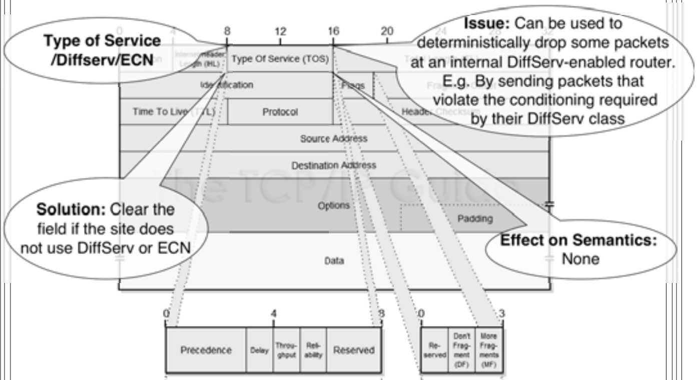
- "Bump in the wire" device that ***normalizes*** the packet stream to remove potential ambiguities before the NIDS monitors it
- "Fail-close" by definition



Normalization Approach

- Normalize IP/TCP layers
- Walk through the packet header of each protocol to be normalized
  - Normalize each field according to standards and internal network
- May not explicitly thwart attacks
  - Still reduces the degrees of freedom to express attacks

IP Normalization Example



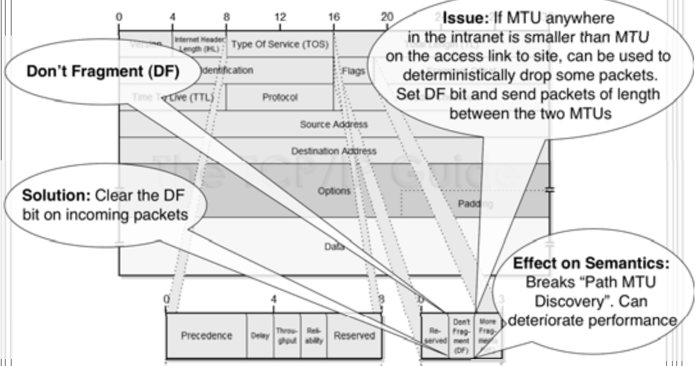
Type of Service /Diffserv/ECN

Issue: Can be used to deterministically drop some packets at an internal DiffServ-enabled router. E.g. By sending packets that violate the conditioning required by their DiffServ class

Solution: Clear the field if the site does not use DiffServ or ECN

Effect on Semantics: None

IP Normalization Examples [contd…]



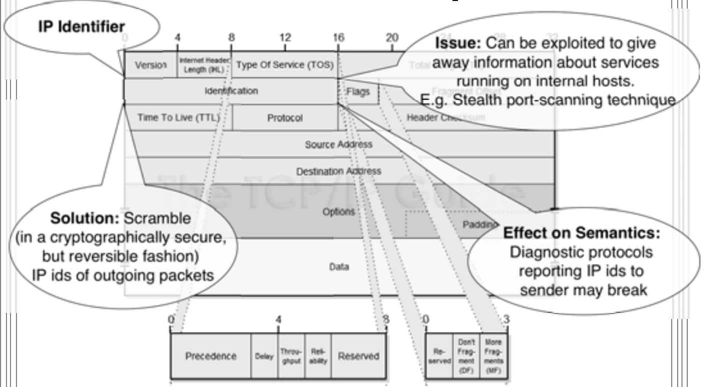
Don't Fragment (DF)

Issue: If MTU anywhere in the intranet is smaller than MTU on the access link to site, can be used to deterministically drop some packets. Set DF bit and send packets of length between the two MTUs

Solution: Clear the DF bit on incoming packets

Effect on Semantics: Breaks "Path MTU Discovery". Can deteriorate performance

IP Normalization Examples [contd…]



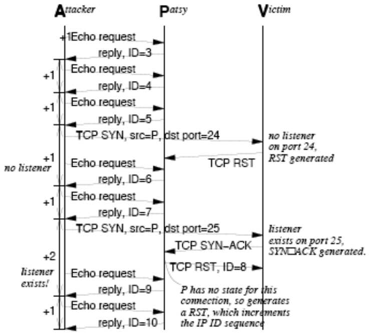
IP Identifier

Issue: Can be exploited to give away information about services running on internal hosts. E.g. Stealth port-scanning technique

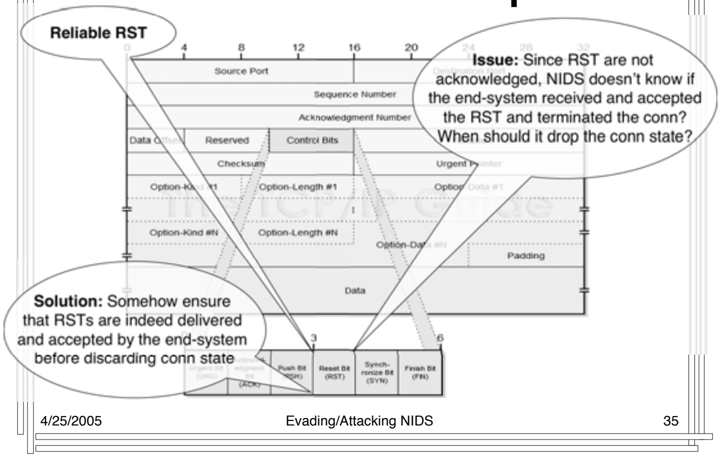
Solution: Scramble (in a cryptographically secure, but reversible fashion) IP ids of outgoing packets

Effect on Semantics: Diagnostic protocols reporting IP ids to sender may break

Stealth Port Scanning



TCP Normalization Examples



Reliable RST

Issue: Since RST are not acknowledged, NIDS doesn't know if the end-system received and accepted the RST and terminated the conn? When should it drop the conn state?

Solution: Somehow ensure that RSTs are indeed delivered and accepted by the end-system before discarding conn state

TCP Normalization Examples [contd…]

- **"Reliable RST" Solution**
  - Retain connection state upon seeing an inbound RST
  - Accompany a Keep-Alive to the end-system with the RST
  - If the end-system received and accepted RST it will send a RST back
  - Else it will send a Keep-Alive back
  - Drop conn state if end-system sends back RST, else retain it

TCP Normalization Examples [contd…]

- **Cold Start for TCP**
  - When to instantiate state for established conn?

- Solution
  - For outbound packets with no known state, instantiate new state
  - For inbound packets with no known state
    - Convert the packet to a Keep-Alive by stripping of data and reducing sequence no.
    - Outbound response to Keep-Alive (if any) will instantiate new state

Incompleteness of Normalization

- Application Level Protocols
  - Cannot be normalized w/o detailed knowledge about them

- Even IP/TCP Level Normalization is Incomplete
  - Handling of TCP urgent pointer depends on the application semantics
  - Socket level options not known to normalizer/NIDS

Normalization Concerns

- End-to-end Semantics
  - Must be preserved for well behaved traffic
  - Sometimes benign traffic may cause ambiguities

- Impact on End-to-end Performance
  - Adversely affects the performance
  - Line-speed operations required

- Normalization vs Protection vs Detection
  - Different from firewalls, NIDS but can share load

Normalization Concerns [contd…]

- State-Holding
  - Needs to hold state to remove reassembly related ambiguities in data flows
  - Performs “triage” (discards state for inactive flows when near resource exhaustion)

- Cold Start Problem
  - How to handle packets for already established connections?
  - Connection characteristics negotiated unavailable

Attacks on Normalizer

- State-Holding Attacks
  - Fragment reassembly
    - Limits memory used for fragments
  - TCP state flooding
    - Limits total memory consumed
  - Not explicitly protecting internal hosts but itself and NIDS by checking memory use

- CPU Overload Attacks
  - Only slows down packet forward rate

Implementation/Results

- ***norm***, a user level normalizer
  - Using commodity PC
  - Large number of normalizations
  - Line speed in bi-directional 100 Mbps env
  - Robust to denial of service attacks
    - Very severe attacks may cause norm to resort to triage

- Kernel level implementations can achieve better results

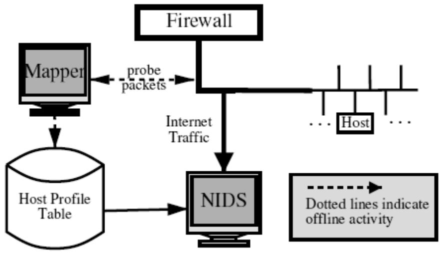
Alternatives to Normalization

- Host-based NIDS
  - Deployment, management issues
- Bifurcating Analysis
  - Fork when ambiguities detected
  - Analyze each possible interpretation
  - Exponential growth in branches (DoS!!)
- **Understand the Intranet**
  - **Particulars of protocol implementations at each end-system, and network segments**

Active Mapping

- Resolves ambiguities without having to intercept or modify the stream
- Acquire sufficient knowledge about the intranet being monitored
  - Make NIDS context sensitive (Bro)
- Use this knowledge to decide if packets will reach the end-systems and their interpretation

Active Mapping Architecture



Mapping Details

- Mapper sits Parallel to NIDS
  - NIDS ignores Mapper traffic to internal hosts
  - NIDS and Mapper can share information
- Mapping done by Sending Probe Packets
  - Service discovery using ICMP echo msgs
  - Hop count and Path MTU discovery
  - Generates host-specific profiles
    - E.g. What policy does the host use for handling IP fragment and TCP segment overlap?

Selected Mappings

- **TCP RST Acceptance**
  - Ideally, accept iff it is within the receiver’s window
- Steps (Repeated with O = 0, 1, W+)
  - Send TCP SYN at Seq No. S
  - Recv SYN-ACK with window W
  - Send ACK to establish conn
  - Send RST at Seq No. S+O
  - Send FIN at Seq No. S
  - Recv one of
    - ACK of FIN --> RST not accepted
    - RST or nothing --> RST accepted

Selected Mappings [contd…]

- **Overlapping IP Fragments**
- Different OSs have different policies
  - BSD Policy
    - Left-trims incoming fragment to existing fragments with lower or equal offset, accepts remaining octets
  - BSD-right Policy
    - Same as BSD, but right-trims
  - Linux Policy
    - Same as BSD, but left-trims only to existing fragments with strictly lower offset
  - First / Last (RFC791) Policies
    - Accepts first/last octet for each offset

Selected Mappings [contd…]

- **Fragment Overlap Handling Example**
  - Data Sent

```
         11
012345678901 --> Higher IP Offset
111 22333     (Fragments 1,2,3)
 4444 555666  (Fragments 4,5,6)
```

  - Data Received

```
111442333666  BSD policy
144422555666  BSD-right policy
111442555666  Linux policy
111422333666  First policy
144442555666  Last/RFC791 policy
```

Selected Mappings [contd…]

- **Overlapping TCP Segments**

- Different OSs have different policies (similar to IP level policies)
  - BSD Policy
  - First Policy
  - Last Policy

Difficult or Intractable Cases

- Application Level Parameters
  - Socket options affecting TCP/IP

- New/Changed Semantics
  - Configuration changes, Upgrades, Patches

- Nondeterministic Packet Drops
  - Drops due to full incoming packet buffer
  - Drops by internal routers (e.g. Diffserv)
  - Drops due to reassembly time-outs

Other Concerns

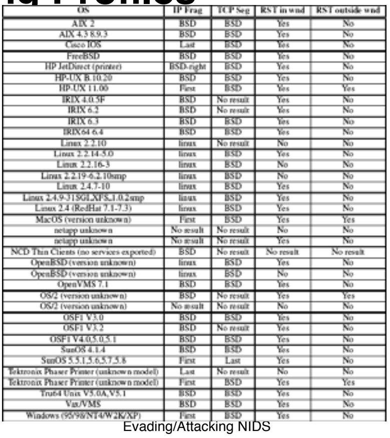
- NAT
  - Mapping becomes difficult, but still possible

- DHCP
  - Integrate DHCP server and Mapper

- TCP Wrappers
  - Host based access control

- Attacks on Active Mapper
  - Firewall traffic to Mapper

Mapping Profiles



OS	IP Frag	TCP Seg	RST in wnd	RST outside wnd
AIX 2	BSD	BSD	Yes	No
AIX 4.3 8.9.3	BSD	BSD	Yes	No
Cisco IOS	Last	BSD	Yes	No
FreeBSD	BSD	BSD	Yes	No
HP JetDirect (printer)	BSD-right	BSD	Yes	No
HP-UX B.10.20	BSD	BSD	Yes	No
HP-UX 11.00	First	BSD	Yes	Yes
IRIX 4.0.5F	BSD	No result	Yes	No
IRIX 6.2	BSD	No result	Yes	No
IRIX 6.3	BSD	BSD	Yes	No
IRIX64 6.4	BSD	BSD	Yes	No
Linux 2.2.10	linux	No result	No	No
Linux 2.2.14-5.0	linux	BSD	Yes	No
Linux 2.2.16-3	linux	BSD	No	No
Linux 2.2.19-6.2.10smp	linux	BSD	No	No
Linux 2.4.7-10	linux	BSD	Yes	No
Linux 2.4.9-31SGI.XFS.1.0.2smp	linux	BSD	Yes	No
Linux 2.4 (RedHat 7.1-7.3)	linux	BSD	Yes	No
MacOS (version unknown)	First	BSD	Yes	Yes
netapp unknown	No result	No result	No	No
netapp unknown	No result	No result	Yes	No
NCD Thin Clients (no services exported)	BSD	No result	No result	No result
OpenBSD (version unknown)	linux	BSD	Yes	No
OpenBSD (version unknown)	linux	BSD	No	No
OpenVMS 7.1	BSD	BSD	Yes	No
OS/2 (version unknown)	BSD	No result	Yes	Yes
OS/2 (version unknown)	No result	No result	No	No
OSF1 V3.0	BSD	BSD	Yes	No
OSF1 V3.2	BSD	No result	Yes	No
OSF1 V4.0,5.0,5.1	BSD	BSD	Yes	No
SunOS 4.1.4	BSD	BSD	Yes	No
SunOS 5.5.1,5.6,5.7,5.8	First	Last	Yes	No
Tektronix Phaser Printer (unknown model)	Last	No result	No	No
Tektronix Phaser Printer (unknown model)	First	BSD	Yes	Yes
Tru64 Unix V5.0A,V5.1	BSD	BSD	Yes	No
Vax/VMS	BSD	BSD	Yes	No
Windows (95/98/NT4/W2K/XP)	First	BSD	Yes	No

Agenda

- Network ID Systems
  - Architecture, Problems
  - Insertion, Evasion, DoS Attacks

- Solutions
  - Traffic Normalization
  - Active Mapping

- **Miscellaneous**
  - **Evasion with Unicode**
  - **Evasion using Polymorphic Shell Code**

Evasion using UNICODE

- Affects string/pattern matching in NIDS Signature Analyzers
- Basic Problems
  - Multiple representations of the same character in earlier UTF-8 standards
    - Current UTF-8 Standard had unique representation
  - Non-compliance to UTF-8 standards by some applications

UTF-8

- Unicode Transformation Format
  - Serializes Unicode code points (U+xxxx) as a sequence of 1-4 bytes
- UTF Extended by a byte
  - Every time the representation got bigger, the earlier transformation formula re-mapped the complete set of previous code points
- Example: '\' character (U+005C)
  - 5C (1B), C19C (2B) and E0819C (3B)

Applications add Complexity

- OS, applications may assign the same interpretation to different code points
- E.g. IIS on Win2K Advanced Server
- No. of different code points for
  - 'A' - 30, 'E' - 34, 'I' - 36, 'O' - 39, 'U' - 58
  - "AEIOU" can be expressed by 83,060,640 different byte streams

Problems Caused

- Multiple representations for characters like '.' and '/' (affect URL/path interpretation)
- No. of signatures required (say, for Snort) explodes exponentially
- NIDS does not know UTF-8 interpretation by end-systems and apps
- Different interpretations by different systems could make it worse

Solutions

- Stick to Unique Interpretations
  - OS and applications should conform to latest UTF-8 standard
- Turn off UTF-8 if not used
  - Works for all mono-lingual sites
- Use Host-based IDS
  - IDS should know the exact interpretation by all apps running on the host

Polymorphic Shell Code

- Basically Code Obfuscation
  - Directory traversal using ".", ".." and "/" are common obfuscation techniques
- Usually employed in Buffer Overflow exploits
- 50 NO-OP instructions on Intel Architecture
  - Increases NIDS's ambiguity problem
- Diff interpretations by diff architectures?

References

- Thomas Ptacek, Timothy Newsham, “Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection”, *Secure Networks*, January 1998.
- M. Handley, V. Paxson, C. Kreibich, "Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics", *Proc. of the 10th USENIX Security Symposium*, 2001.
- Umesh Shankar, Vern Paxson, "Active Mapping: Resisting NIDS Evasion Without Altering Traffic." *Proc. of the 2003 IEEE Symposium on Security and Privacy*, May 2003.
- IDS Evasion Techniques and Tactics
  http://www.securityfocus.com/infocus/1577
- IDS Evasion with Unicode
  http://www.securityfocus.com/infocus/1232
- What is polymorphic shell code and what can it do?
  http://www.sans.org/resources/idfaq/polymorphic_shell.php

Why are the attackers mostly feminine (“she ..”) ??