Covert Channels in TCP and IP Headers

Drew Hintz

drew@overt.org

http://guh.nu
At A Glance

• Who are you? Who are they?
• Definitions
• Analysis of some covert channels
• Attack against timestamp covert channel
• Detection and prevention
• New covert channel tool
Who Are You?

- Need to communicate covertly, but typical encryption is not the complete solution.

- Why not just encrypt?
  - Encryption may be outlawed
  - Key escrow may be mandatory
    - US & UK Government
  - Might raise too many eyebrows
    - Thrown in jail in China
    - Employer may question what you are doing
    - ‘Malicious’ code command and control
Who are “they”?

• Casual Observer
  – Watching everyone
  – Automated systems sifting on keywords
  – Can only keep minimal state on traffic

• Dedicated Observer
  – Looking specifically at a few suspect people
  – All portions of traffic closely monitored
  – Many resources available, can easily keep state
How covert is covert enough?

• Semi-Covert: Fooling the Casual Observer
  – If examined, traffic appears unusual
  – Assumes “they” won’t bother looking
  – Detecting it requires a low to moderate amount of work

• Truly Covert: Fooling Everyone
  – Traffic appears completely normal
  – Will work even if “they” know the procedure used
  – Detecting it is as hard as breaking the underlying cryptography
Two Types of Covert Channels

- Extra communication to a host
- Hiding the fact that you’re even communicating with a host
Extra Communication to a Host

• Useful when you must hide the fact that you’re encrypting data

• Method
  – Uses some amount of cover/permissible traffic
  – Sender embeds covert message outbound
  – Receiver gets traffic as normal
  – Receiver analyzes traffic, retrieves message
Finding a Good Covert Channel

• First find a place that random data is being transmitted naturally
  – Ex. Initial Sequence Numbers, complex timing of network transmissions
  – Then replace that random data with your own ‘random data’ which is actually an encrypted message
A simple example

- Alice wants to send a message to Bob
- Alice FTPs Bob a couple of old vacation pictures, meanwhile Bob records all traffic
- Alice encodes the secret message byte by byte in the padding of several TCP segment headers
- Bob looks at padding of recorded traffic
Unseen Path of Communication

• Useful if you do not want your association with a node to be known
  – Communicating with a closely scrutinized node
  – Accessing ‘forbidden’ material
  – Malicious activity

• Use another node to relay information for you
Finding a Good Channel

• Use a hard to monitor node as an intermediary

• Protection is possible even if all networks are watched
  – Mixes, onion routing
    • Prevents association of incoming and outgoing
  – Also use extra communication covert channel
    • Prevents detection that a node is relaying info
Unseen Path Example

- Alice and Bob both are allowed to make requests to the same small webserver
- To transmit a one, Alice pounds the webserver with heavy traffic
- To transmit a zero, Alice doesn’t make any requests
- Bob makes requests to the webserver and measures latencies
Evaluating a Covert Channel

• Bandwidth
  – bits per TCP connection
  – bits per packet
• Ease of Detection
• Permissibility
  – How often will it be permitted?
• Prevention
• Difficulty of Implementing
• Special Cases or Restrictions
What are we going to look at?

• Extra communication covert channels
  – Needed if encryption is restricted
  – Hidden path is being looked at heavily by anonymity researchers

• TCP/IP Headers
  – Occurs frequently on networks
  – Able to piggy-back on legitimate traffic
TCP URG Pointer

- Sequence number that points to end of urgent data
- Only interpreted if URG control bit is set
- Set the 16-bit URG pointer to the value that you’re transmitting, however do not set the URG bit
- To be a bit more stealthy, the URG pointer value should be restricted to be near the sequence numbers so that it actually could be pointing at something
URG Pointer Evaluation

• Bandwidth
  – Good: 16 bits per TCP Segment; much less if the URG pointer is restricted to be near sequence numbers

• Detection
  – Easy: URG pointer rarely used, and should never be used without setting the URG control bit
  – Also, URG pointer must actually point to data

• Prevention
  – Moderate/Easy, if URG bit is not set, rewrite pointer
  – Perhaps disallow use of URG
URG Pointer Eval (continued)

• Permissibility
  – Traffic normalizers and some firewalls may clear the URG pointer, especially if the URG control bit is zero

• Difficulty of Implementing
  – Easy, simply replace bits in packets in transit

• Special Cases
  – Can not be used in TCP segments where the URG pointer is actually being used
Padding & Reserved bits

- Similar to URG example
- Lower bandwidth
- Padding is easy to detect, because almost always set to zeros
- Padding and reserved bits may be rewritten by some routers
IP Type of Service

• Indicates quality of service requested
  – Precedence, delay, throughput, reliability, reserved
• Set the IP ToS byte to your data
• To be much more stealthy, only modulate the Delay bit
IP ToS Evaluation

• Bandwidth
  – 1 byte per IP datagram if using entire ToS field
  – 1 bit per IP datagram if using only delay bit

• Detection
  – Easy if using entire ToS field, because the entire field is never used
  – Moderate if using only the delay bit
    • Look for frequent occurrences of set delay bits
ToS Evaluation (continued)

- **Permissibility**
  - Passed through tested equipment

- **Prevention**
  - Easy, rewrite all ToS bits
  - Could slightly alter handling of traffic

- **Difficulty of Implementing**
  - Easy, simply replace bits in packets in transit

- **Special Cases**
  - Could slightly alter actual handling of traffic, but not noticeable
Initial Sequence Number (ISN)

- Sequence numbers used to index TCP data being transmitted
- ISN should be random to prevent TCP session hijacking and spoofing
- Choose your initial sequence numbers to be the message to be transmitted
ISN Evaluation

• Bandwidth
  – Low: 32 bits per TCP connection

• Detection - ‘Impossible’

• Prevention
  – Difficult, have to proxy all TCP connections
ISN Evaluation (continued)

• Permissibility
  – Will pass through all, except some proxies

• Difficulty of Implementing
  – Moderately Easy, simply replace the function used to generate initial sequence numbers

• Special Cases
  – Some OSs (such as Windows 98) do not choose random ISNs
Timestamp Low-bit Modulation

- TCP option
- Modulate low bit of TCP timestamp to convey data
- Presented by Giffin, et al at PET2002
- At low bandwidths, the low bit of the timestamp is quite random (based on complex timings)
Timestamp Evaluation

• Bandwidth
  – Low: one bit per TCP segment

• Detection
  – Very, very difficult for low bandwidth

• Prevention
  – Moderate/Easy, strip out TCP timestamp option

• Permissibility
  – Permitted on just about all networks
Timestamp Eval (continued)

• Difficulty of Implementing
  – Moderate, many things must be kept in mind
  – Must be sure timestamps are monotonically increasing
  – In order to do this, a fast connection will be slowed down while sending covert data
Timestamp Detection - Fast

• When on a fast connection, the sending of TCP segments will be slowed down to a fixed rate

• Algorithm to detect:
  – Count number of different & total timestamps sent by a particular host
  – Calculate the ratio of total to different timestamps
  – If timestamp covert channel is being used on a sufficiently fast network, the ratio will converge to about 1.94

• To prevent slow down more than needed
Timestamp Detection - Slow

- For even a slow connection, it is very difficult, but possible, to detect
- The covert channel makes the low bit more random than it normally is
- Algorithm:
  - Record all the low bits of the timestamp
  - Put them through a complex randomness test
  - If very random, then covert channel being used
- To prevent introduce some non-random data
Detection and Prevention

- Detect anomalous traffic
  - Some IDSs can do this, but it can be very noisy
- Perform normalization of traffic
  - norm
  - BSD pf
  - Use a pump method to defeat timing channels
- Can not close all covert channels, only possible to decrease bandwidth and ease
Implementation Issues

- Encryption must happen at some level
- Is data being transmitted?
- What data is being transmitted?
- Reliability vs. Bandwidth
Encryption is a Key Component

- Good encryption must be used for theoretically secure covert channels
- Assume covert channel method is well known
- Must ensure same ciphertext is not transmitted multiple times
  - Stream cipher initialized with a common time and key could be used
Is data being transmitted?

- Send checksum at end of each sequence of data
  - Receiver examines traffic and attempts to find data/checksum matches
- Checksum must be securely keyed, otherwise attacker could do the same
- Magic flag that receiver watches for
What Data is being Transmitted?

- Can use unmodified portion of header and data as a nonce
- Use the hash of the nonce as an index into the data being transmitted
- Transmit data sequentially
- Implement more advanced protocols
Reliability vs. Bandwidth

• Using hash of nonce provides high reliability, low bandwidth, simple solution

• Transmitting data sequentially provides high bandwidth, low reliability, simple solution

• Implementing a more advanced protocol is ideal for long messages
Covert Channel Tool

• Sending component is a proc Linux Kernel Module that modifies outgoing TCP/IP traffic by replacing hard_start_xmit
• Receiving component sniffs incoming traffic using libpcap
Implemented Covert Channels

- Initial Sequence Number
- TCP Timestamp low-bit modulation
  - Has high-speed protection
- Urgent Pointer
- IP Type of Service
- TCP Reserved Bits
Implemented Data Indexing

- Sequential transmission of the data
- SHA of unmodified portions of header and data used as index into the data
Work to be Done

- Improve user interface
- Built-in encryption
- Add more options for covert channels
- Analyze how various routers, IDSs, etc. handle illegal data
- Implement more robust protocol for data transmission (TCP over TCP?) (:
Any Questions?

• Get the goods:

http://guh.nu/projects/cc/

• Shameless plug for OSVDB
  – Vulnerability database by the community
  – http://www.osvdb.org