



Enhanced TCP SYN Attack Detection

Authors: Vrizzlynn L. L. Thing,
Morris Sloman,
Naranker Dulay

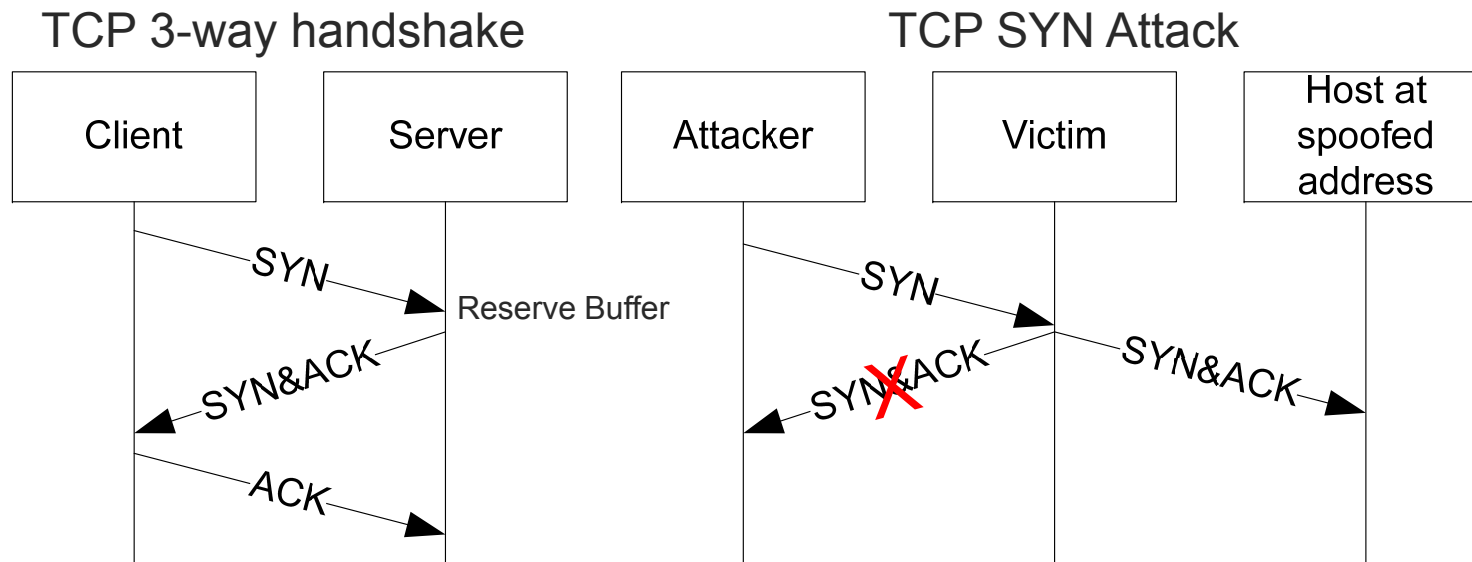
Email: vlt@doc.ic.ac.uk

Date: 5th November 2007

[Overview]

- Problem definition
- Analysis of existing detection mechanisms
 - SYN-FIN/RST (H. Wang et al., Jun 2002)
 - SYN-Dog (H. Wang et al., Jul 2002)
- Our New Bot Buddy SYN Attack
- Our Solution
- Analysis and Evaluation
- Results

[Problem]



- Attacker spoofs Source IP address of request
- Victim reserves resources for connection request and sends response to spoofed address
- No acknowledgement received to complete handshake

[SYN-FIN/RST Detection]

- Based on inherent TCP SYN-FIN/RST symmetry
- Detection at the attackers' and victims' ends
- Monitor SYN, FIN (for normal close) and RST (for active abort) packets in both directions

Number of SYN \approx Number of FIN + RST

- Simple counting without address or flow matching
- Abrupt positive fluctuation in the difference between SYN and FIN/RST detected by non-parametric CUSUM algorithm

[SYN-FIN/RST Detection]

- Other reasons for generating RST packets:
 - 1) arrival of data packets for which no connection has been established
 - 2) arrival of TCP segments with inappropriate sequence numbers
 - 3) arrival of SYN&ACK packets for which no SYN has been initiated
 - 4) arrival of TCP packets for closed ports
- Assumes 75% of RST packets as valid i.e. from aborts, which weakens the mechanism

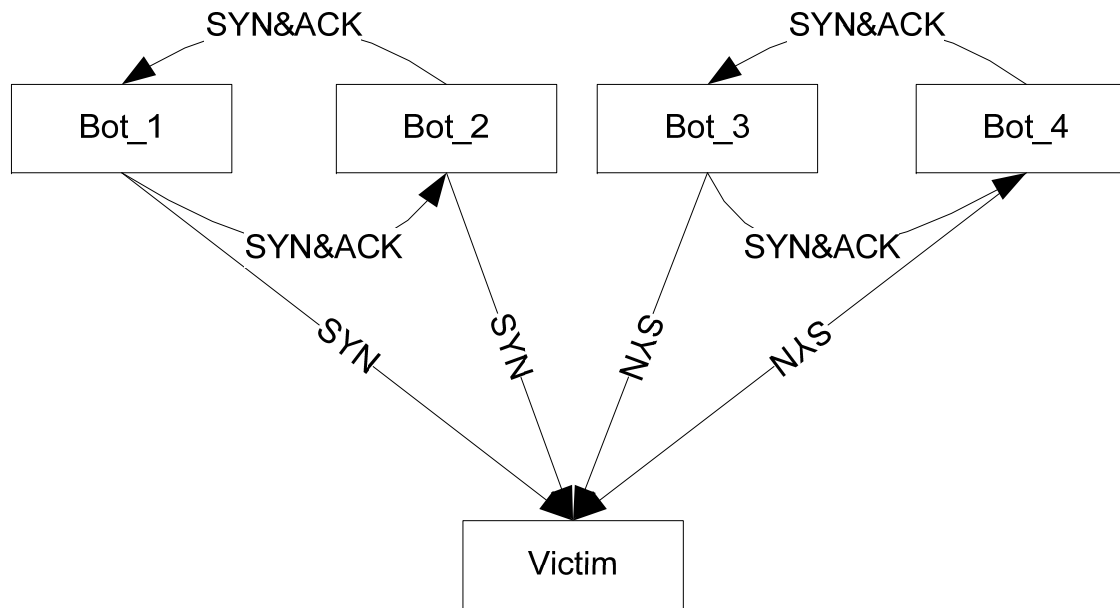
Invalid RST Packets

- During SYN attack, attackers target random ports as indicated in our previous studies.
 - Victim server generates RST packets due to (4)
- Attackers use address spoofing
 - SYN&ACK packets generated by victim server would be delivered to hosts located at spoofed addresses, triggering replies of RST packets to victim due to (3) and (4)
- Higher numbers of RST could result in attacks not being detected
- New variation of attacks could send a mixture of SYN and FIN/RST packets to defeat the detection mechanism

[SYN-Dog Detection]

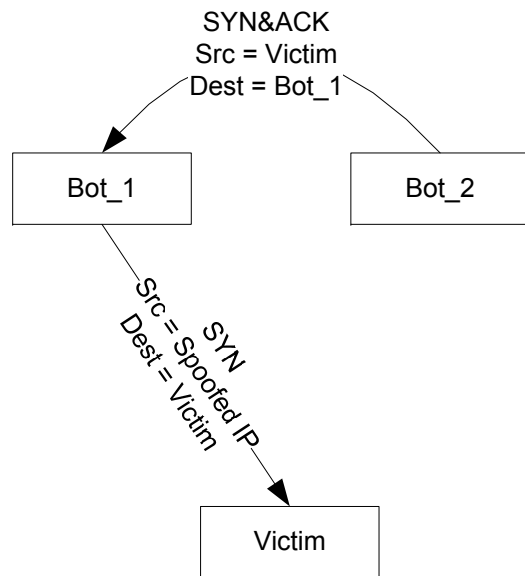
- Based on inherent TCP SYN-SYN&ACK symmetry
- Implementation and detection at the “source”
- Agent monitors difference of number of outgoing SYN and incoming SYN&ACK packets
- Uses non-parametric Cumulative Sum approach to detect abrupt rise in the difference
- New variation of attacks with mixture of SYN and SYN&ACK packets has no effect on detection mechanism

[Bot Buddy SYN Attack]



- Our suggested “new variation of co-ordinated attack”
- Easily implementable by attackers due to bot master having information of all bots within the botnet
- Defeats the SYN-Dog mechanism

[Our Solution]



Attack packets' header

- Perform SYN-SYN&ACK pair matching using addresses
 - Incorporate Bloom filter to achieve space and time efficiency
1. Bloom filter, $F[0\dots m-1]$, as an m -bit array which is initialized to 0
 2. Define each element to be stored in the filter as e_{out} (similar to incoming e_{in}):

$$e_{out} = src \parallel dest \parallel R$$

3. k hash functions, $h_1() \dots h_k()$, are used to compute k key values for e_{out} to be stored in the filter:

$$F[h_i(e_{out}) \bmod m] = 1, \text{ for } i = 0 \text{ to } k$$

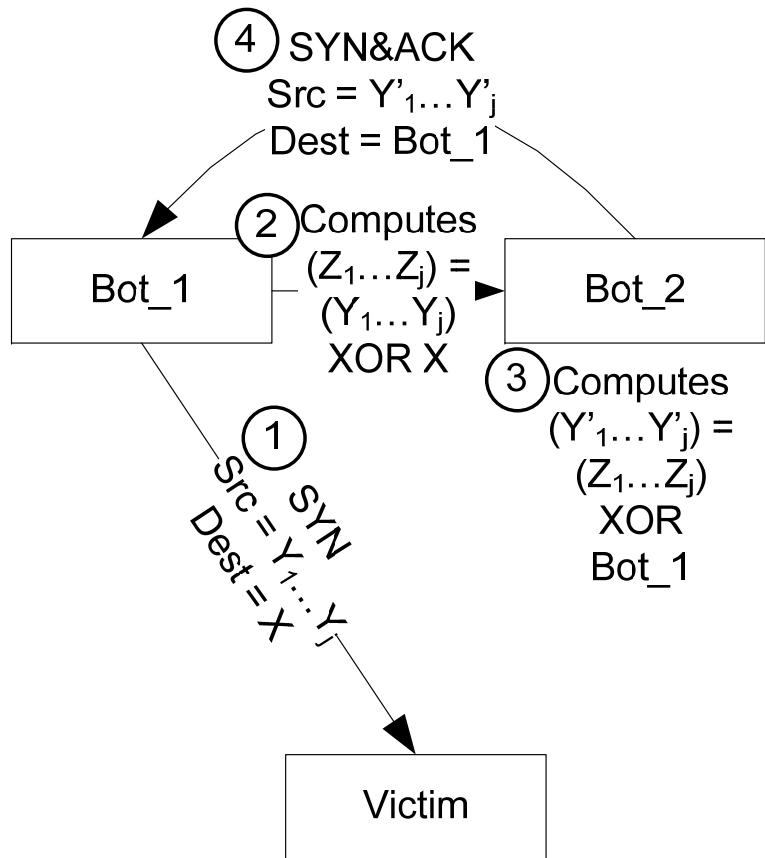
4. To be counted as a valid SYN&ACK packet for inclusion into the detection algorithm, all the bits at positions,

$$\{h_i(e_{in}) \bmod m\}$$

in the filter array must be set to 1.

[Our Solution]

$$e_{out} = src \parallel dest \quad \text{instead of} \quad e_{out} = src \oplus dest$$



- Assuming the algorithm used by the routers is known
- $X \text{ XOR } Y = Z$ implies that knowing Z and X , Y can be recovered
- Bot_1 only needs to send either Y or Z to Bot_2
- Bot_2 can calculate source addresses Y' such that $Y' \text{ XOR } Bot_1 = Y \text{ XOR } X$ (Victim's address) and sends: SYN&ACK with Src = Y' and Dest = Bot_1
- Router then sees these SYN&ACKs as valid as the hash matches

[Analysis]

- Time required to store and search for an element is a fixed constant, $O(k)$
- Probability that a bit in the array is not set:

$$p_0 = \left(1 - \frac{1}{m}\right)^{kn}$$

- Probability that a bit is set:

$$p_1 = 1 - p_0$$

- Probability of a false positive error (each of the k array positions computed for the element must be set to 1):

$$p_e = (p_1)^k$$

- Probability of collision (with real source address):

$$p_c = \frac{1}{2^{32}}$$

- Number of SYN&ACK errors during the Bot Buddy SYN Attack:

$$Err_{t,att} = \sum_{i=1}^{SYN_{t,att}} (2p_c + (1 - p_c)(p_e))$$

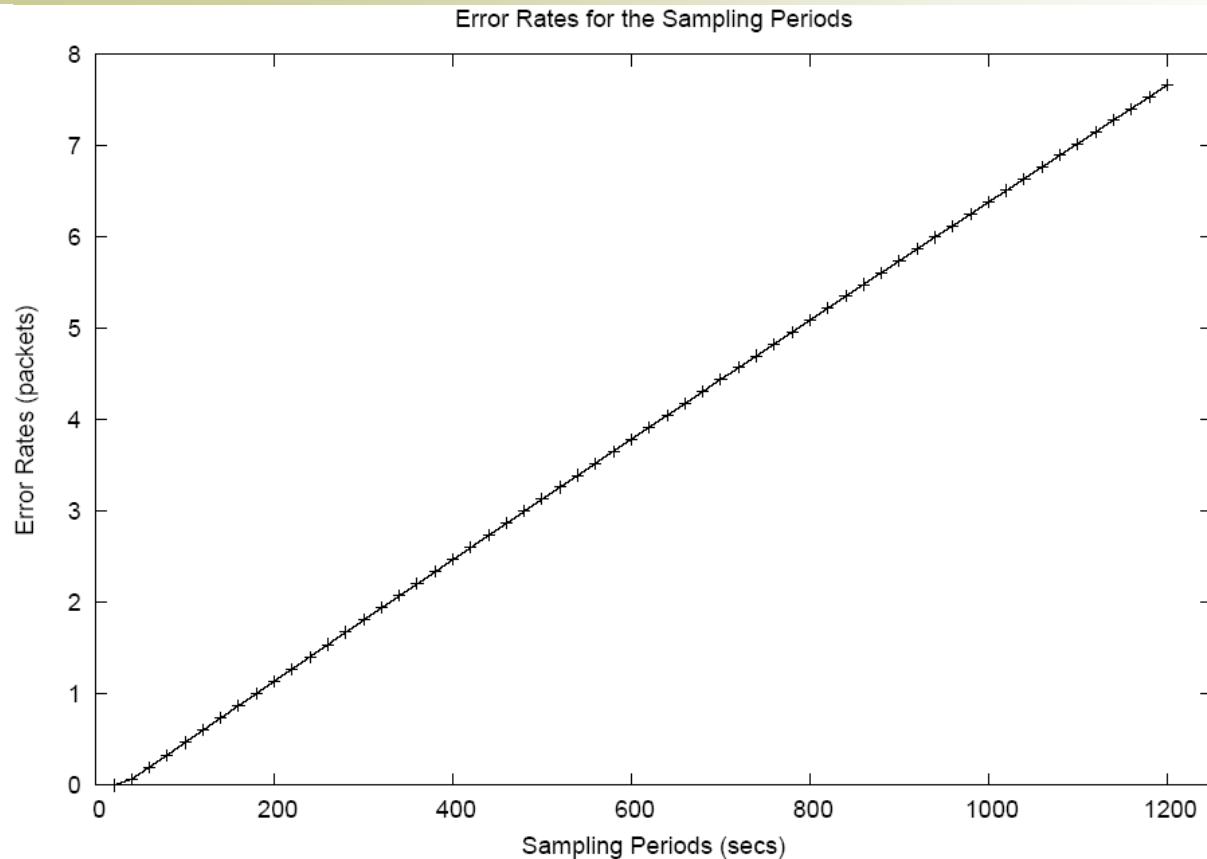
- Difference between outgoing SYN and incoming SYN&ACK (during interval t):

$$\Delta_t = SYN_{t,norm} - SYN \& ACK_{t,norm} + SYN_{t,att} - Err_{t,att}$$

Evaluation

- Using the experiment parameters in the SYN-Dog paper:
 1. Sampling period = 20 secs
 2. Attack rate = 60 SYN packets/sec
 3. Normal traffic traces obtained from University of North Carolina (monitored on high-speed OC-12, 622Mbps link connecting to the rest of the world) show:
 - $\text{SYN}_{t,\text{norm}} \approx 1200$ to 1900,
 - $\text{SYN\&ACK}_{t,\text{norm}} \approx 1050$ to 1700,
 - $\Delta_{t,\text{norm}} \approx 200$ in 10-sec intervals, consistently
 4. Our mechanism's parameters:
 - $k = 1$ (i.e. 1 hash function)
 - $m = 4,194,304$ (i.e. Bloom filter size of 512KB)
 5. Assumption:
 - Each SYN&ACK for each SYN sent out would take an average of 150ms to arrive at the leaf router of the attack bot,
 - $\text{SYN}_{t,\text{norm}}$ averaged to 155 packets/sec,
 - $\Delta_{t,\text{norm}} = 400$ packets per sampling period,
 - Attack starts one sampling period after the legitimate traffic

[Results]



After 20 minutes of attack, the incorrectly validated SYN&ACKs is just 8, among the 70,800 Bot Buddy SYN&ACKs received. Therefore, the SYN flood in addition to the Bot Buddy SYN Attack are successfully detected.

Conclusions

- Analysed both stateless SYN-SYN&ACK and SYN-FIN/RST detection mechanisms
- Discovered inherent vulnerability of SYN-FIN/RST detection mechanism caused by RST packet count
- Both mechanisms suffered in terms of reliability in view of new variations of TCP attacks, e.g. failing to detect our Bot Buddy SYN Attack
- Proposed an enhanced detection mechanism incorporating Bloom filter to handle the attack
- Analysed and evaluated our mechanism, finding it to work effectively to detect both conventional SYN attack and new Bot Buddy SYN Attack

[Hashing Overhead]

- Hash functions satisfy the 2 basic properties: 1) one-way, irreversible and 2) collision resistant
- Implementation using CryptoPP library and SHA1 hashing algorithm
- Performance benchmark on an Intel Core 2 1.83 GHz processor (Windows XP SP2 in 32-bit mode) is shown by the CryptoPP developers to be 155MB/sec
- Assuming R to be 32 bits, it translates to a hashing speed of 13544106 hashes/sec or $7.38e-8$ secs/hash in our mechanism
- For SHA256 and SHA512, it will take $1.41e-7$ and $1.15e-7$ secs/hash, respectively