Protecting the Infrastructure

Diego Montenegro Andrew Park Bobby Vellanki

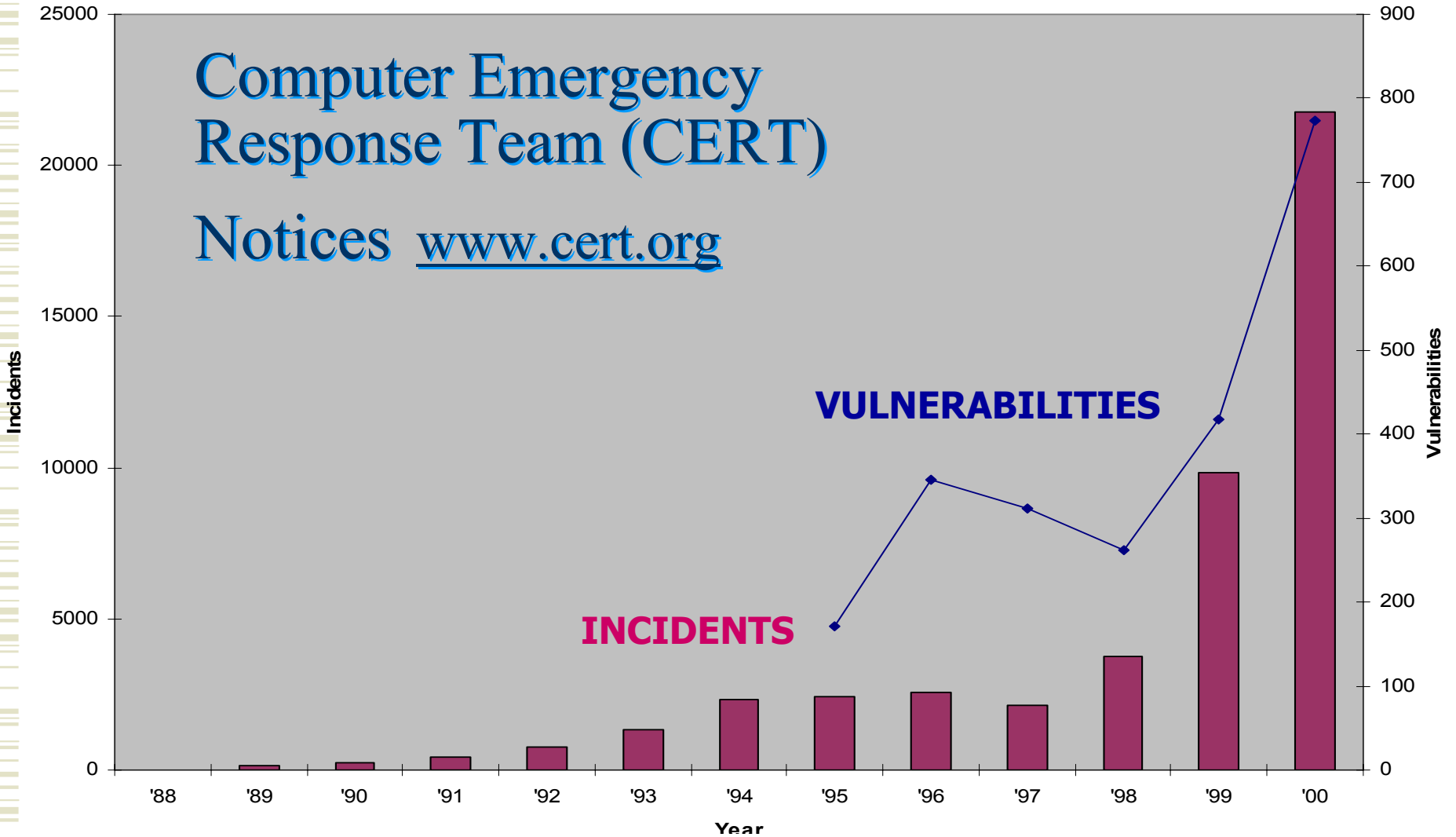
Protecting the Infrastructure

- Problems
  - Denial of Service (DoS)
- Black Box Properties
- Solutions :
  - Reactive : IP traceback
  - Proactive : SoS
- Open Problems

Protecting the Infrastructure

- 1991 – 1994 had 298% increase in the number of computer intrusions and a 702% increase in the number of sites affected.
- US Laws against new types of computer crime can either be analogized to traditional crimes or new specific laws can be created
- The “Computer Fraud and Abuse Act” was enacted in 1984 and revised in 1994
- Onel de Guzman, suspected author of the “Love Bug” virus (May 2000), was never prosecuted because virus dissemination was not then a crime in the Philippines

Protecting the Infrastructure



Protecting the Infrastructure

Moonlight Maze (March 1998) – Suspected Russian intelligence stole secret information from NASA, the Pentagon, and other government agencies

(USA Today Electronic News, 10 Oct 2001)

"For 3 ½ years, a shadowy group of computer hackers has broken into hundreds of computer networks and stolen thousands of top-secret files on Pentagon war-planning systems and NASA technical research. Dubbed the "Moonlight Maze" group, the hackers continue to elude the FBI, the CIA and the National Security Agency, despite the biggest cyber probe ever. And while no one knows what is being done with the classified information, some fear the thefts may be the work of terrorists or that the information could be sold to terrorists."

Denial of Service Attacks

"An explicit attempt by attackers to prevent legitimate users of a service from using that service" (CERT)

There are three basic types of DoS attacks:

1. Consumption of scarce, limited, or non-renewable resources
2. Destruction or alteration of configuration information
3. Physical destruction or alteration of network components

Consumption of Scarce Resources

Basic ways to consume resources:

- Network Connectivity
- Using ones own resources against them
- Bandwidth Consumption
- Consumption of Other Resources

Consumption of Scarce Resources

Network Connectivity: Prevent hosts or networks from communicating on the network.

"SYN flood" – The attacker establishes "half-opened" connections causing a denial of legitimate connections

Client	Server	
SYN---------->		Attacker would use IP spoofing
<----------SYN-ACK		No general solution but likelihood of an attack can be decreased by using filters
ACK---------->		

Blaster Worm – windowsupdate.com

Consumption of Scarce Resources

Using own resources against them: When a connection is established between two UDP services, each of which produces output, then these two services can produce a very high number of packets

UDP Packet Storm – By connecting a host's chargen service to the echo service on the same or another machine, all affected machines may be effectively taken out of service because of the excessively high number of packets produced

Solution – Remove chargen and echo services

Consumption of Scarce Resources

Bandwidth Consumption: An attacker can consume all the available bandwidth on ones network by generating a large number of packets directed to their network

Typically, these packets are ICMP ECHO packets but in principle they may be anything

The attacker does not need to be operating from a single machine; he may be able to coordinate several machines on different networks to achieve the same effect

Consumption of Scarce Resources

Consumption of other resources:

• Many systems have a limited number of data structures available to hold process information. An attacker can implant a simple program that just makes copies of itself

• Consuming disk space by generating excessive numbers of mail messages, intentionally generating errors that must be logged, placing files in network shares

• Using “lockout”

Other DoS Attacks

Destruction of configuration information - An intruder may be able to alter or destroy configuration information that prevents you from using your computer or network. Some examples include Windows registry alteration or changing routing information in routers

Physical Destruction or Alteration of Network Components
- Physically damaging hardware components

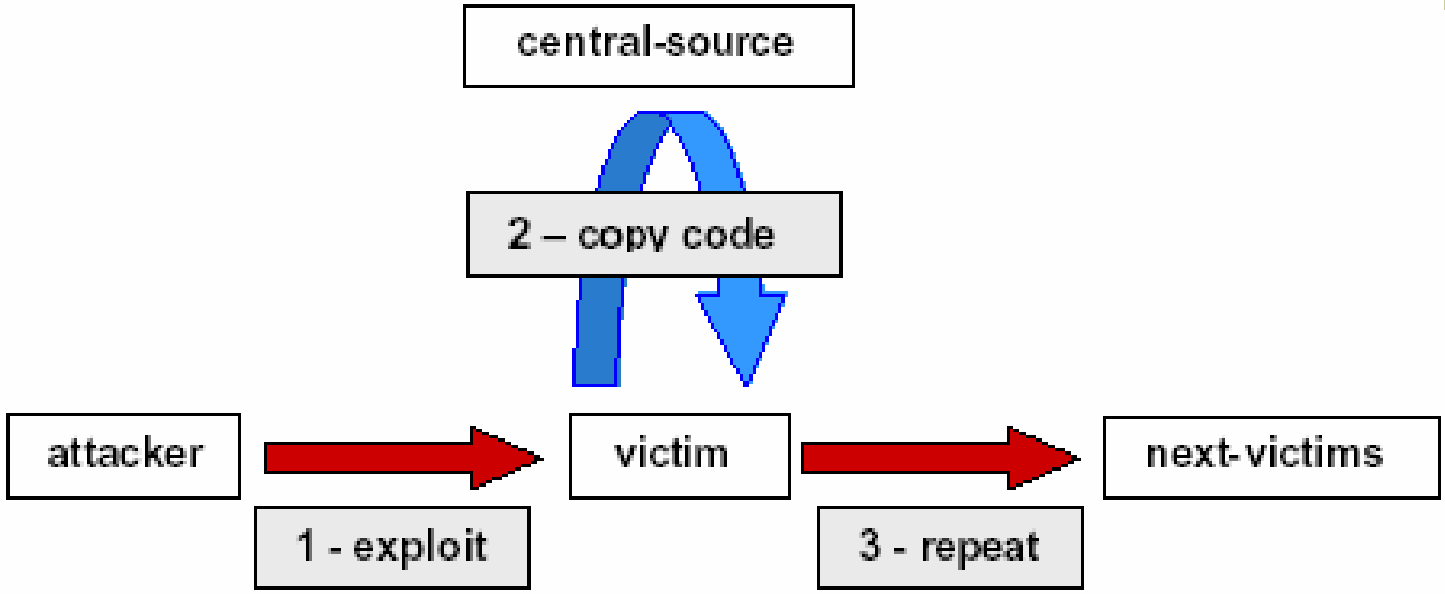
Automation

DoS attacks are becoming automated:

- Central source propagation
- Back-chaining propagation
- Autonomous propagation

Worms create copies of themselves by spreading out onto other machines on the network and implanting viruses that harm each machine that the occupy

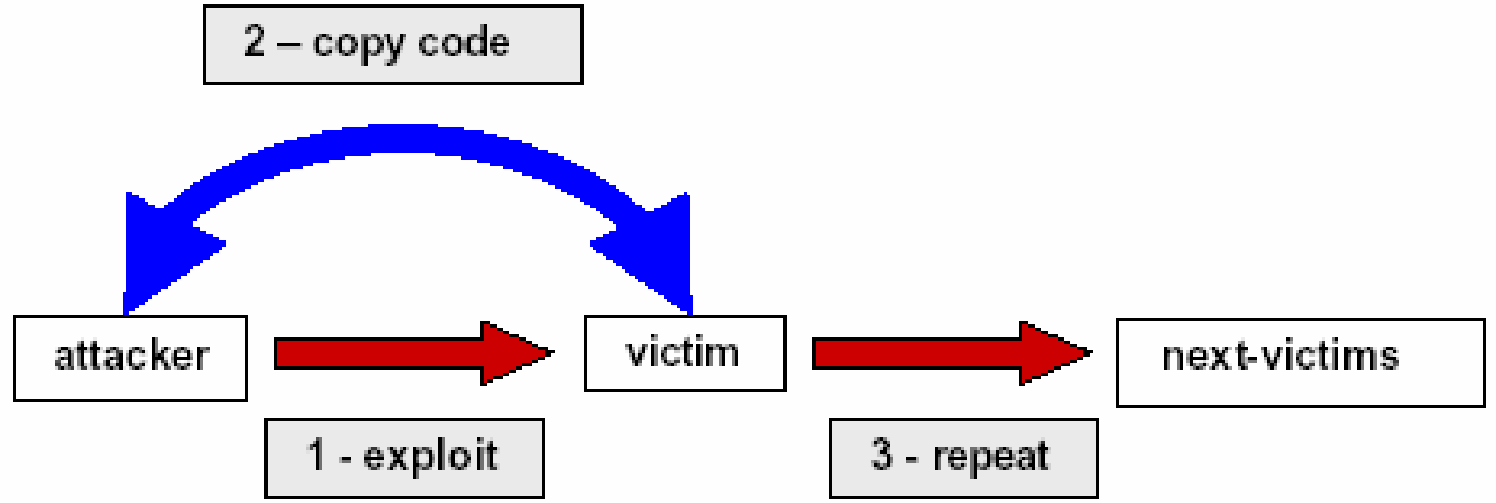
Automation



Central source propagation – Scripts copy code from a centralized location to the victims system first then sends the worm onto the next victim

1i0n worm

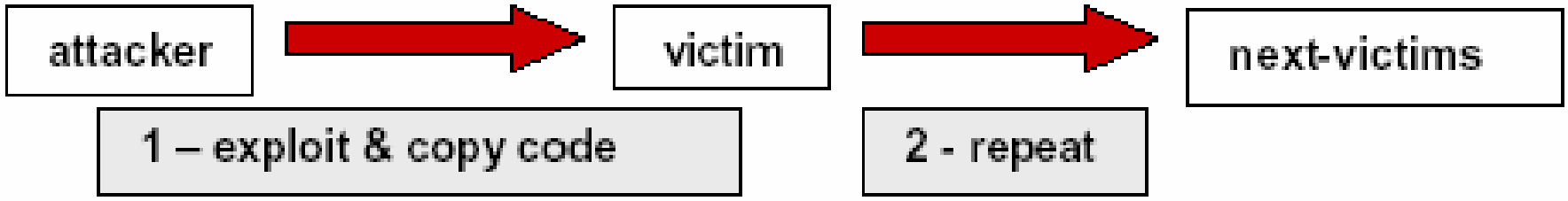
Automation



Back-chaining propagation – The worm is copied from the previous host and repeated. This method is more survivable because there is no single point of failure

Ramen worm

Automation



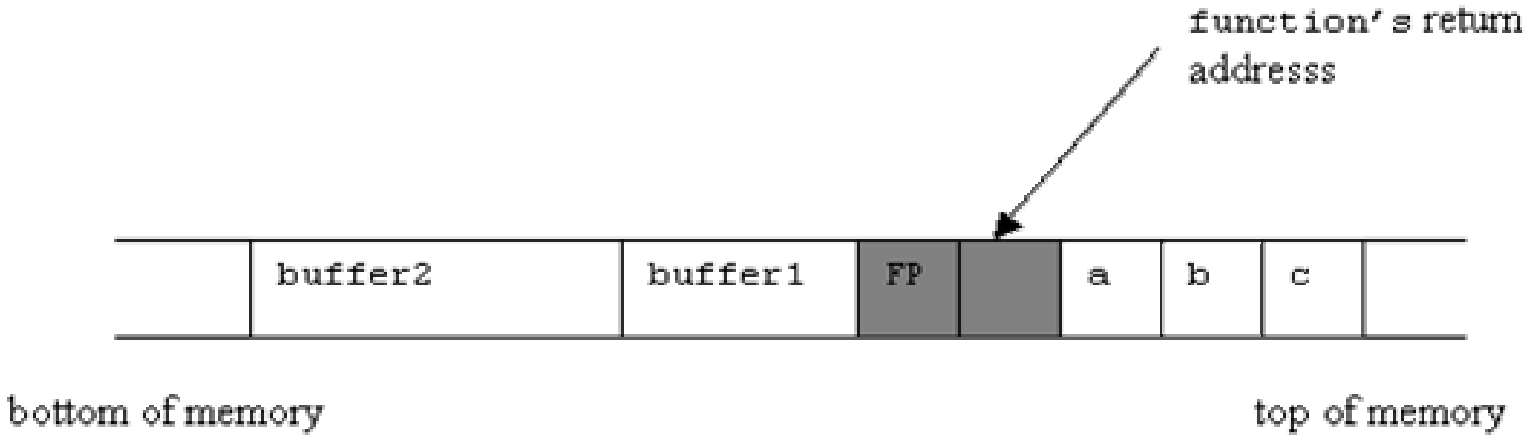
Autonomous propagation – Similar to the previous method but the steps of exploiting the victim and copying of the code is done in one step as to avoid file retrieval

Code Red and Morris worm of 1988

Buffer Overflow Problem

```
void function(char *str)        int main() {
char buffer[16];                char *str = "I am greater than 16 bytes";
strcpy(buffer,str); }           function(str); }
```

The extra bytes run past the buffer and overwrites the space allocated for the FP, return address and so on. Thus, the flow of the program can be modified to move to an undesirable location



Code Red (CRv1 and CRv2)

- Code Red Version1 (July 12 2001)

  -Exploited a buffer overflow problem with Microsoft's IIS webservers

  - Generates a random list of IP addresses for future victims but the seed was static so all infected computers used the same list

  - Launch a DoS attack against www.whitehouse.gov

  - Resides in memory so simply restarting the computer was enough to disinfect it but was likely to be re-infected because of the same IP address list being created

  - Spread slowly and did minor damage

Code Red (CRv1 and CRv2)

- Code Red Version2 (July 19 2001)

  - All computers that did not use Code Red v1 patch where potential victims

  -Very similar to CRv1 with the slight modification of creating a random list of IP addresses

  - Infected more than 359,000 computers in approximately fourteen hours and 43% of the infected hosts were in the US

  - Routers, switches, DSL modems, printers, and other devices would crash or reboot when an infected machine attempted to send them a copy of the worm

Spread of Code Red v2

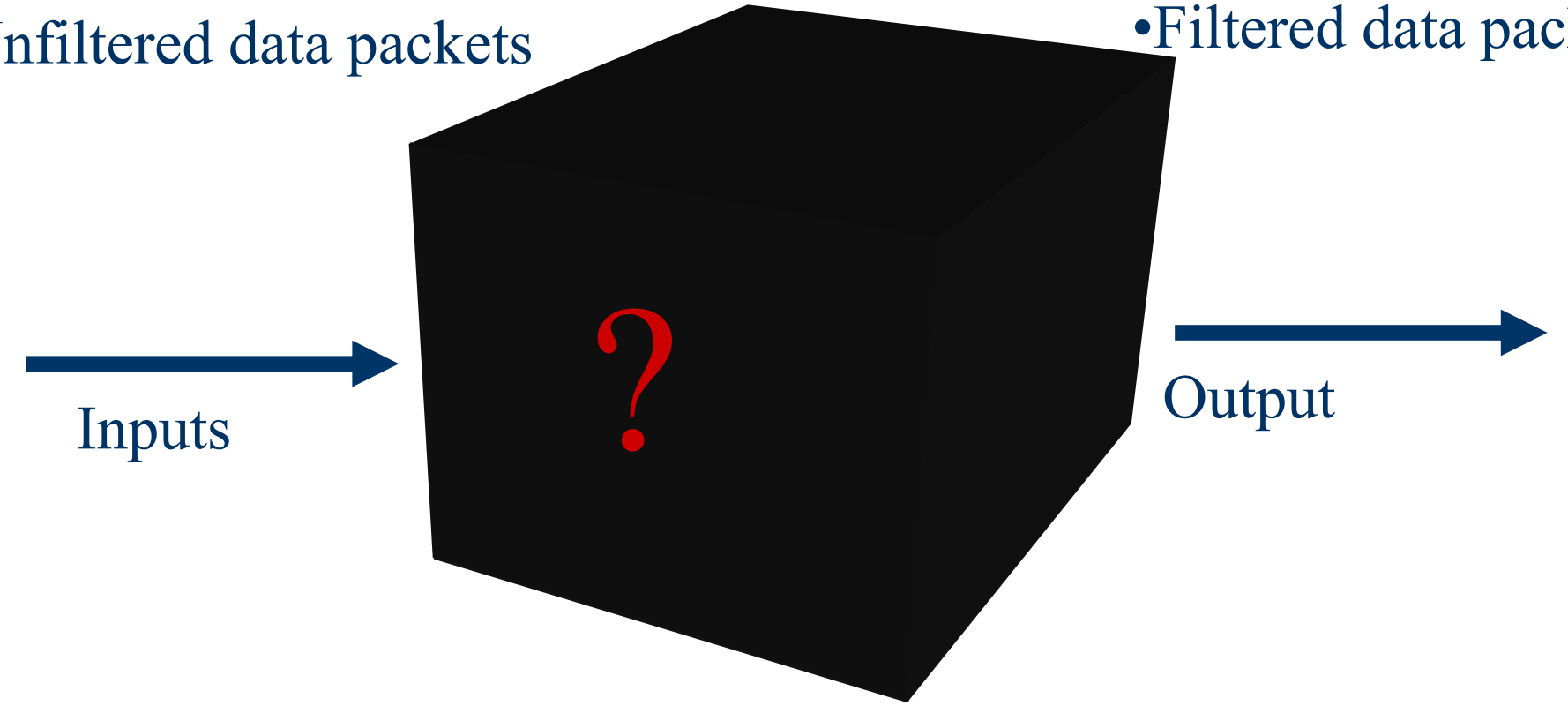
Picture was removed

Lost productivity (~$1.5billion)

Black Box Properties

- Vulnerable networks
- Unfiltered data packets

- Secure network
- Filtered data packets



Black Box Properties

- Reliable
- Time-Efficient
- Cost-Efficient
- Robust
- Scalable
- Secure

IP Traceback

- DOS attacks consume resources of remote hosts or networks
- No way to tell if a packet is forged
- Tough to trace packet back to source (no state)
- No entity is responsible to ensure correct source address in IP protocol
- Tradeoff: routing vs. tracking

IP Traceback

Assumptions/Properties:

- Packets may be addressed to more than one physical host
- Duplicate packets may exist in the network
- Routers may be subverted
- Attackers are aware they are being traced
- Packet size should not grow as a result of tracing
- End hosts may be resource constrained
- Traceback is an infrequent operation

IP Traceback

Approaches:

1. **Ingress Filtering**
2. **Link Testing**
   - **Input Debugging**
   - **Controlled Flooding**
3. **Logging**
4. **ICMP traceback**
5. **Packet Marking**

- Some store the state and perform computations at the end hosts while others use resources only within the network.

IP Traceback

1. **Ingress Filtering**
   - Configure routers to block packets that arrive from illegitimate source addresses
   - Requires routers to examine the source address of every packet
   - All routers must participate (dependent upon peers)
   - Potential overhead
   - No protection on transit networks
   - Attackers may forge source addresses
   - Summary: Improves Internet's robustness to DOS attacks but not foolproof
   - Most routers employ this technique

IP Traceback

2. Link Testing

- Start with closest routers and interactively test upstream routers
- Assumes attackers are still active

a. Input Debugging

- Allows an operator to determine which incoming port the packet arrived on by filtering packets
- Victim develops attack signature and the operator installs the debugging filter on the upstream egress port. (repeated recursively)
- Reveals which upstream router originated traffic
- **Cons:** Lots of overhead and ISP's don't have any economic incentive. Requires support from network operators.

IP Traceback

b. Controlled Flooding

- Doesn’t require network operators
- Tests links by flooding routers and observing how this effects the attacks.
- Flood upstream routers and loaded links will drop packets.
- **Cons:**
  - It is a DOS itself
  - Requires knowledge of the network topology
  - Doesn’t work for distributed DOS

IP Traceback

3. Logging

- Log packets at routers and use datamining techniques to find path
- An *attack graph* is constructed from a set of attack paths
- Able to trace path even after the attacks have stopped
- **Cons:**
  - Requires a lot of resources
  - Database integration between providers (no incentive)
  - May have false graphs if routers are subverted
  - No commercial organizations use this approach

IP Traceback

4. ICMP Traceback

- Similar to Packet Marking
- Every router copies a sample packet (1/20,000) and adjacent router info. into a special ICMP Traceback messages.
- Victim can use these messages to reconstruct the path
- Assumes “many” packets are sent
- **Cons:**
  - Requires input debugging (not available in all routers)
  - Requires key distribution to prevent false messages
  - Relies on input debugging capability (not available on all routers)

IP Traceback

5. Packet Marking:

- Can be used after the attacker “leaves”
- Consists of two parts: **Marking Procedure** and **Path Reconstruction Procedure**
- Assumes there are thousands or millions of packets
- Assume it is rare for packets to follow different paths in a short period

Packet Marking Algorithms:

A. Node Append
B. Node Sampling
C. Edge Sampling

IP Traceback

A. Node Append

- Append each node's address at the end of the packet

Pros:

- Simple, robust and easy to converge

Cons:

- High overhead of appending data
- May not have sufficient space in the packet
- Leads to fragmentation
- Easily forgeable

IP Traceback

B. Node Sampling

- Sample one node at a time instead of the whole path (“node” field)
- Each router writes its address with probablity *p*
- The victim will have received at least one sample from each router (assume large number of packets)
- Construct the router order
- If $p > 0.5$, it will be robust against single attacker. (No way for the attacker to insert “false” router by contributing more samples than the downstream router)

IP Traceback

B. Node Sampling (cont.)

Cons:

- Getting total router order is a slow process (e.g. if d=15 and p=.51, receiver needs 42,000 packets on avg.)
- Routers far away from the victim contribute small samples (especially if *p* is high)
- Not robust against multiple attackers. (If both attackers are at the same distance away, they will both be sampled with the same probability)

IP Traceback

C. Edge Sampling

- Encode edges instead of individual nodes
- Two address fields (start, end) and a distance field
- Each router writes it's address with a probability of *p*
- If a router decides to mark a packet, it writes down its own address in the start field with dist = 0. If the dist = 0 already, write its address in the end field.
- If a router decides not to mark the packet, it will increment distance by 1.
- Optimal P: $p <= 1/d$

IP Traceback

	Management overhead	Network overhead	Router overhead	Distributed capability	Post-mortem capability	Preventative/ reactive
Ingress filtering	Moderate	Low	Moderate	N/A	N/A	Preventative
Link testing						
Input debugging	High	Low	High	Good	Poor	Reactive
Controlled flooding	Low	High	Low	Poor	Poor	Reactive
Logging	High	Low	High	Excellent	Excellent	Reactive
ICMP Traceback	Low	Low	Low	Good	Excellent	Reactive
Marking	Low	Low	Low	Good	Excellent	Reactive

IP Traceback

Single Packet Traceback

- Most attacks (DOS) assume there are large number of packets
- Routers and Operating Systems can be disabled by a single packet attack. (e.g. the Teardrop attack crashes MS windows with 1 packet)
- Not possible to determine the attacker of a single packet given the previous algorithms.
- Possible Solution: SPIE (Source Path Isolation Engine)

IP Traceback

SPIE

- Uses auditing techniques to support traceback of individual packets
- Stores the packet digests
  - Reduced storage requirements
  - Preserves traffic confidentiality
  - Packet content used as input must uniquely represent an IP packet
  - Must be collision-free
  - 24 invariant bytes of the packet (16-IP header, 8-payload)

IP Traceback

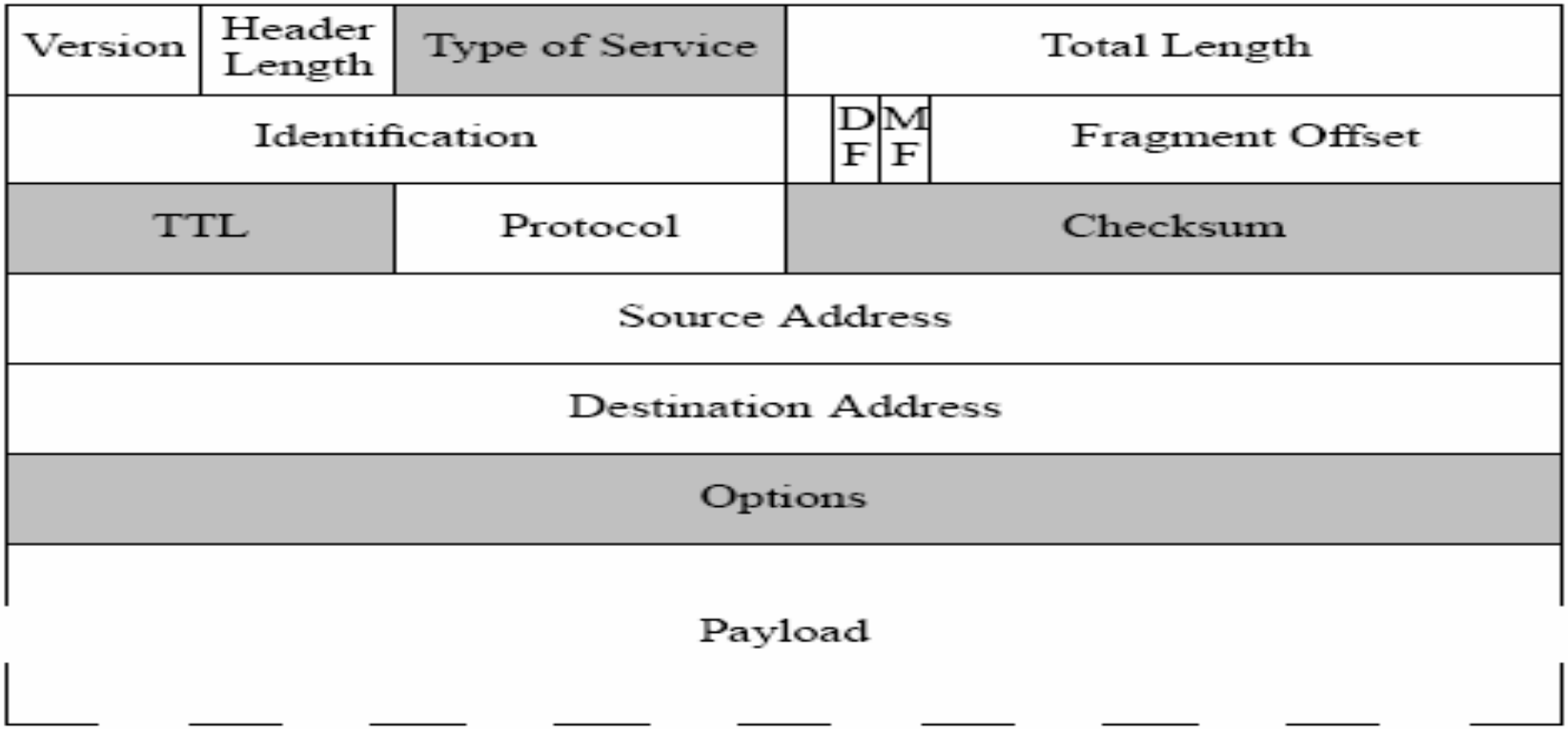


Fig. 1. The fields of an IP packet. Fields in gray are masked out before digesting, including the Type of Service, Time to Live (TTL), IP checksum, and IP options fields.

IP Traceback

SPIE

- Packet auditing, query processing and attack graph generation are dispersed among separate components
- SPIE-enhanced routers maintain a cache of packet digests. If a packet is determined to be offensive, SPIE queries routers for packet digests. The query results are used to simulate reverse-path flooding algorithm to build an attack graph.
- Each router has a DGA (Data Generation Agent)

IP Traceback

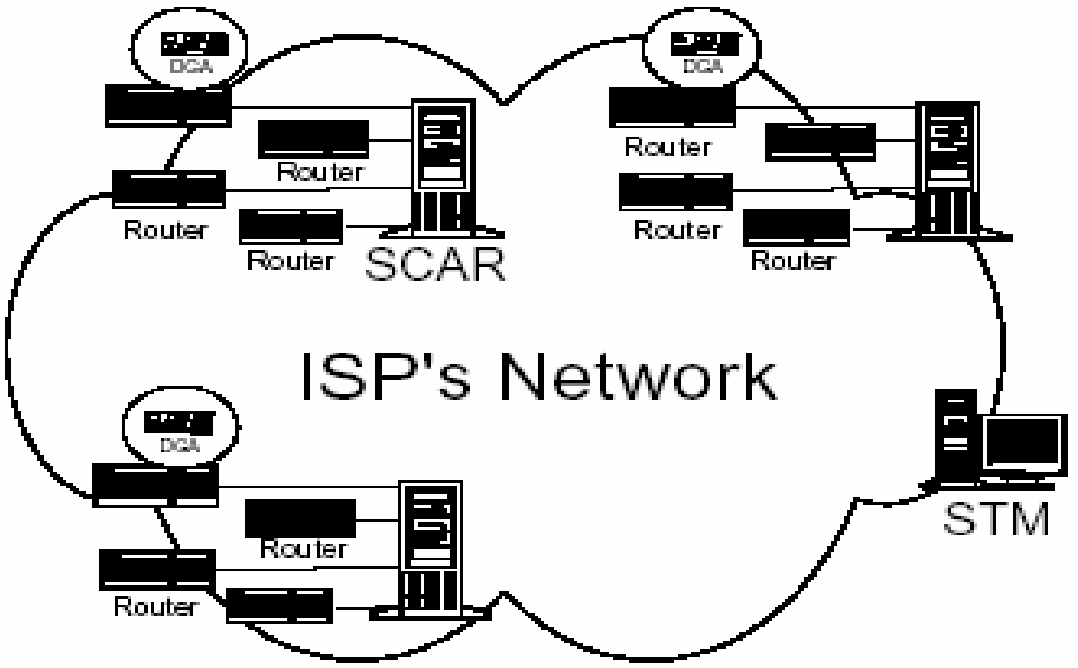


Fig. 4. The SPIE network infrastructure, consisting of Data Generation Agents (DGAs), SPIE Collection and Reduction Agents (SCARs), and a SPIE Traceback Manager (STM).

IP Traceback

Conclusions

- Most of these algorithms are useless unless all the routers participate
- Attack path must be found in a timely fashion
- Storing state vs. Testing upstream

SOS: Secure Overlay Services

Protecting Against DoS attacks:

- Reactive: Wait for an attack to be launched before taking appropriate measures to protect the network.

- ProActive: Eliminate all possibility of becoming a target by aggressively filtering and blocking all incoming packets whose source addresses are not "approved".

SOS: Secure Overlay Services

Why not Reactive?

- Methods that filter traffic by looking for known attack patterns or statistical anomalies in traffic patterns can be defeated by changing the attack pattern and masking the anomalies that are sought by the filter.

- Since the Internet spans multiple administrative domains and legal jurisdictions, it is very difficult to shut down an attack by contacting the administrator or the authorities closest to the source.

- Even if possible, it is often the case that the source of the attack is not the real culprit but simply a node that has been remotely subverted by a cracker.

SOS: General Info

- Geared toward supporting Emergency Services or similar Types of communication.

- Architecture is constructed using a combination of secure overlay tunneling, routing via consistent hashing and filtering.

- Addresses the problem of secure communication between a pre-determined location and users located anywhere in the wide-area network, who have authorization to communicate with that location.

- Main focus is on sites that store information that is difficult to replicate due to security concerns or due to its dynamic nature.

SOS: Main Principles

The two Main Principles behind the design are:

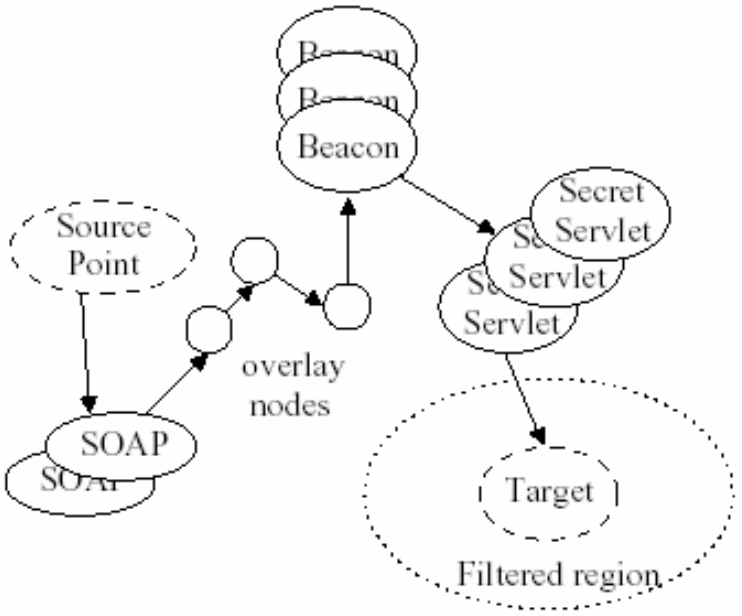
1- Elimination of communication “pinch” points, via combination of filtering and overlay routing to obscure the identities of the site whose traffic is permitted to pass through the filter.

2- The ability to recover from random or induced failures within the forwarding infrastructure or within the secure overlay nodes.

SOS: Architecture

- SOS is a network overlay, composed of nodes that communicate with one another atop the underlying network substrate. These are known to the public and in consequence, to the attackers.

The Basic SOS Architecture



SOS: Filtering

- Knowledge of the target IP allows an attacker to bomb the target location with packets.
- To prevent this, a filter can be constructed that drops illegitimate packets at some point in the network.
- For this protocol, it is assumed that the filter can be constructed so that attackers do not have access to routers inside the filtered region.

SOS: Filtering

- Legitimate users can reach the target by setting the filter around the target to permit only those IP addresses that contain legitimate users. There are 2 problems with this:
  - Legitimate user moves, changes IP or ceases to be legitimate
  - Illegitimate user spoofs the source address of its transmissions to be that of a known legitimate user.

Solution:

. Target selects a Subset of nodes Ns to act as *Forwarding Proxies*.

SOS: Filtering

- Filter is set to allow packets only from overlay nodes $n \in Ns$.
- An attacker with knowledge of the Proxies IP's can still launch 2 forms of attacks:
  - Attack the target by spoofing the Proxy's IP address.
  - Attack the Proxy itself, to cut off communication.
- Solution: Hide the identities of the Proxies. These hidden proxies are known as *Secret Servlets*.

SOS: Reaching the Servlets

- To activate a Secret Servlet, the target sends a message to an overlay node that it chooses, informing it of its task.

- The dynamic nature and the high level connectivity that exists when routing atop a network overlay, allows to construct a routing mechanism that will route to a destination, while utilizing minimal amount of information about the identity of the destination.

SOS: Reaching the Servlets

- This is used to complicate the job of an attacker by making it more difficult to determine the path taken within the overlay to a secret servlet.

- In addition, it is easy to recover from a breach in communication due to attacks that shut down a subset of overlay nodes.

SOS: Connecting to the Overlay

- Not all legitimate users reside at nodes that participate in SOS.

- A *SOAP (Secure Overlay Access Point)* is a node that will receive packets that have not yet been verified as legitimate and perform the verification.

- IPsec is used for this task, because it supports secure exchange of packets by encrypting both the header and the payload of the packet.

SOS: Connecting to the Overlay

- Having a large number of overlay nodes to act as SOAPs increases the bandwidth resources that an attacker must obtain to prevent legitimate traffic from accessing the overlay.

- SOS becomes a large distributed firewall that discriminates between authorized and unauthorized traffic.

SOS: Routing through the Overlay

- Having each overlay participant select the next node at random is sufficient to eventually reach a Secret Servlet. However, this is very inefficient; the expected number of intermediate nodes being contacted is O(N/Ns).

- The paper proposes a routing algorithm based on the Chord Service, in which, with only one additional node knowing the identity of the Secret Servlet, the route from a SOAP to the Servlet has an expected path length of O(log N).

SOS: Routing through the Overlay

- Chord is a Peer-to-Peer lookup service that uses hashing to map an arbitrary identifier to a unique destination. Each overlay node maintains a list that contains O(log N) identities of other nodes.

- Given the destination identifier, each node knows how to choose a member in its list such that, from an arbitrarily chosen starting node, the destination node to which the identifier hashes is reached in O(log N) overlay hops.

SOS: Routing through the Overlay

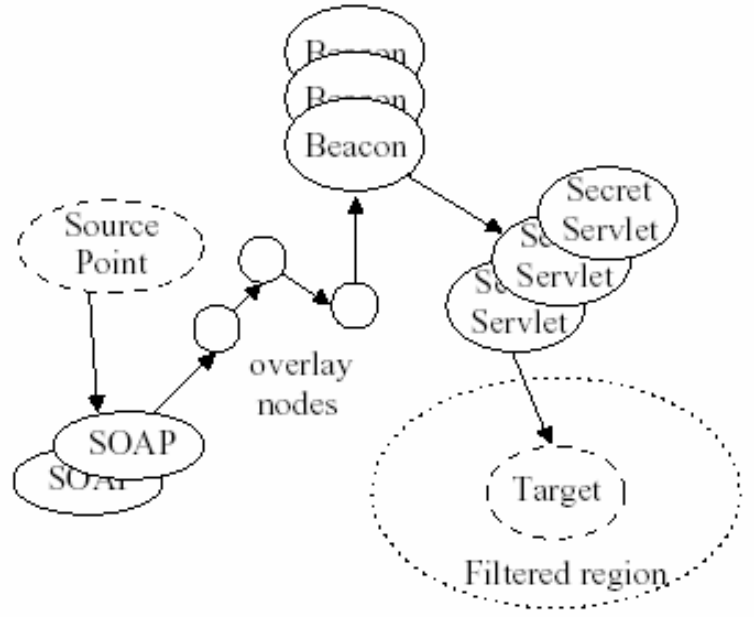
- In SOS, the identifier used is the IP address of the target. Thus, Chord can be used to direct a packet from any node in the overlay to the node that the identifier is mapped to. This node to which Chord delivers the packet is not the target, nor is it necessarily the Servlet. This node is called the *beacon*.

- When a packet is approved by a SOAP for forwarding over the overlay, the hash on the IP address of the target is used as the key.

- The last step is to reveal the Secret Servlet's identity to the beacon. This is achieved also using Chord.

SOS: Routing through the Overlay

- By providing only the beacon with the identity of the secret servlet, the packet can be delivered from any SOAP to the target, by travelling across the overlay to the beacon, to the secret servlet and finally to the target.



SOS: Redundancy

- Having a single SOAP, beacon or Secret Servlet weakens the SOS architecture, in that a successful attack on any one of these nodes can prevent legitimate traffic from reaching the target. Fortunately each component is easily replicated and furthermore, an attack to any of these components, after found, can be easily repaired.

SOS: Robustness

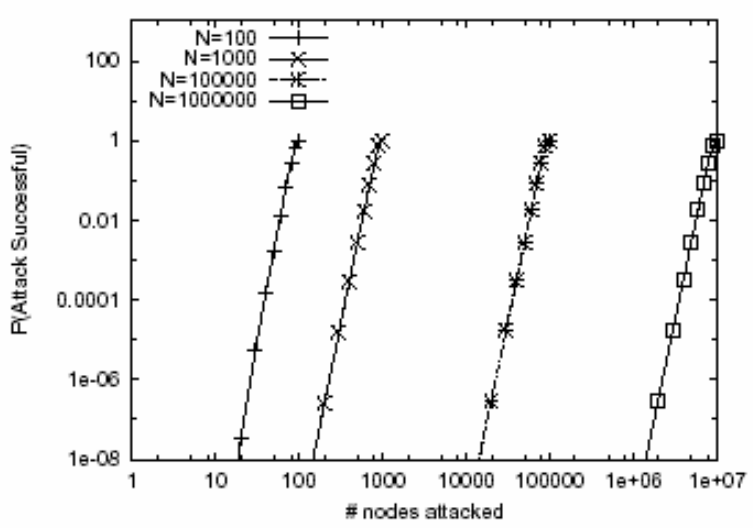
- Why is this robust against DoS Attacks?

1. If any access point is attacked, the confirmed source point can simply choose an alternate access point to enter the overlay.

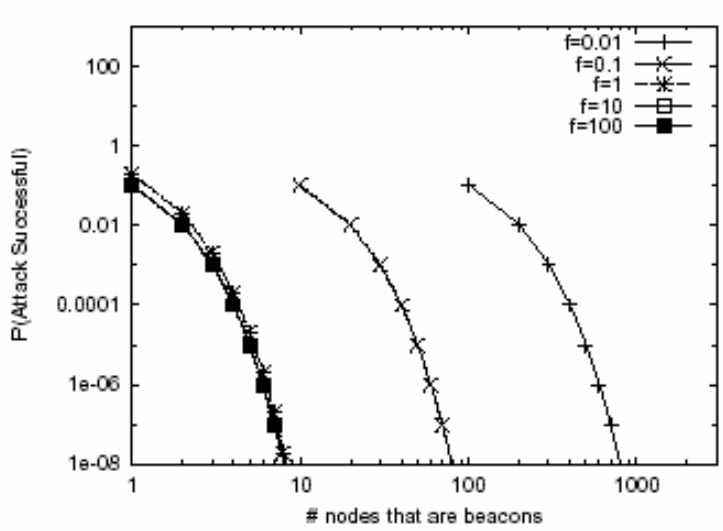
2. If a node within the overlay is attacked, the node simply exits the overlay and the Chord service self-heals, providing new paths over the re-formed overlay to beacons.

3. If a Secret Servlet's identity is discovered and the servlet is targeted as an attack point, or attacks arrive at the target with the source IP address of some secret servlet, the target can choose an alternate set of secret servlets.

SOS: Performance



(a) Varying number of attackers and nodes in the overlay



(b) Varying number of beacons and secret servlets

SOS: Implementation

The SOS architecture can be implemented using existing software and standardized protocols, making its adoption and eventual use, easier.

- **Filtering**: all high and medium range routers (price and performance), as well as most OS, offer some high-speed packet classification scheme that can be used to implement the the target perimeter filtering.
- **Authentication and Authorization**: practically all commercial and free OS include an implementation of IPsec.

SOS: Implementation

Tunneling: once traffic is inside the overlay, it must be routed towards the beacons. This can be accomplished using standard traffic tunneling techniques, like IP-in-IP encapsulation, GRE encapsulation or IPsec in tunneling mode. The routing decisions inside the overlay are based on a Chord-like mechanism.

SOS: Discussion

There are still some open problems to be discussed in SOS, like:

- **Attacks from inside the overlay**: the paper assumes that no malicious user can successfully bypass the protection perimeter. What happens if it is possible, due to security management oversights or development bugs?
- **Scalability**: The architecture presented allows communication from a single confirmed source point to a single target. What are the problems that arise when trying to scale the protocol to handle numerous confirmed source points transmitting to multiple targets?

SOS: Discussion

Timely Delivery: To achieve security, SOS forces traffic through a series of overlay points that perform different tasks. The latency that occurs is far from minimal (10 times larger than direct communication). Can we create “shortcuts” that do not compromise security and allow timely delivery?