The Spread Identity (SI) Paradigm : Applications in the Internet and Beyond

Dhananjay S. Phatak

University of Maryland Baltimore County (UMBC)

phatak@umbc.edu

Original version : 2004

Revisions: 2005, 2008, 2011, and November 2012

Parts covered by US patent # 7,853,680 B2: issued 14th Dec 2010

More patents are pending

SI: Exec Summary (first 3 viewgraphs)

- Inspired by spectacular success of Spread-Spectrum @ L1
- @L3 (network layer) identity of a host = its IP address

∴ SI dynamically {assigns/(de-assigns=kills)} self IP addrs

Unique capabilities enabled by SI :

1. Dynamically issued *dest-addr* ≡ flow-marker ⇒
   - Ultrafast detection of abnormal behavior by simple "token matching" at the destination router/gateway, which in-turn enables fast detection of
     - bot-nets
     - DDoS attacks in progress or impending (being planned)
     - Many types of intrusion as well as exfiltration attempts to leak information

Unique capabilities cont…

2. Enables multi-level, multi-pronged, scalable, robust and highly effective defenses as well as *offenses* against all types of DDoS attacks
3. Enables new robust network traceback capabilities
4. Completely resolves the address-scarcity problem (ex: in IPv4, or more generally, in any name-space)
5. Enables novel, robust anonymity mechanisms
6. All changes are confined to perimeter gateways which we call "SI gateways" (SIGs)
   - All upstream Infrastructure is left untouched
   - Substantial installed-base of operating systems running internal hosts, as well as the application suites, is also left untouched

⇒ fully backward compatible and incrementally deployable

Advantages ++

7. (source , destination) address pairs are leveraged as dynamic {flow-markers (and/or) access-control tokens}, which makes monitoring/processing of flows very easy, which in turn substantially simplifies the control plane
8. Significantly improves the overall security
9. All the above benefits accrue simultaneously in addition to the conventional (prior-known) advantages of indirection (ex: improved support for host-mobility, dynamic load-balancing, …. etc.)
10. The Meta concept of "Spreading Identity" is applicable in many scenarios beyond communications

Sounds too good to be true? read on ….

Topics

- Evolution of SI paradigm as a natural culmination of prior work on Dynamic Transport Selection (DTS)
- The SI Internet communications architecture
  - First version : end-to-end and clumsy
- Commercial viability : ($$$+++ or **NULL)?

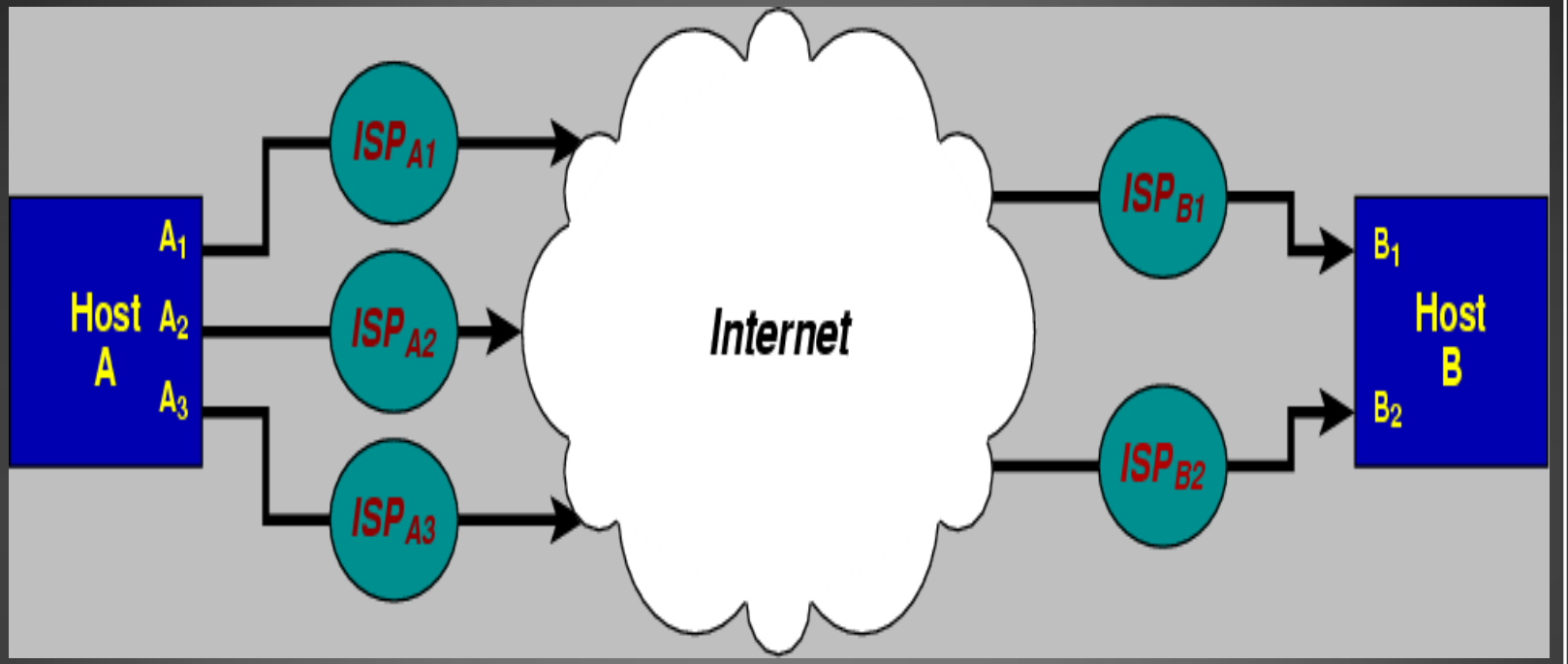
A brief history

- Evolved out of the "Dynamic Transport Selection" (**DTS**) project (2002 — 04), which was supported in part by Aether Systems and National Science Foundation (NSF)
- First version of SI**:** conceived in 2004; published in 2005 in the IEEE Securecomm'05 ref [1].
  - first version **:** end-to-end and clumsy
- Drastic improvements in 2006, 2007 including the crucial novel idea of "pooling" addresses at the perimeter SI Gateway and dynamically assigning them traffic at any level of granularity including{sessions, flows, connections, transmit-windows, packets}
- US Patent application filed in 2007
- First patent issued on 14th Dec. 2010, more pats pending...
- Comprehensive publication in *The Journal of Computer Security Vol. 22, No. 2, pp 233—281, March 2013.*

DTS Motivation

- Rapid expansion of wireless technologies ⇒ multiple transport conduits: 802.x LANS, PANs (Bluetooth…), Wide area GPRS, cellular-phone modems
- DTS problem: select "*k*" best out of "*n*" network interfaces available to a node so as to optimize a given set of objectives including {bandwidth (b/w) aggregation, jitter control, congestion, power used, total $ cost ….etc.} over a specified time interval
- Turns out to be a dynamic Pareto-optimization problem
- Illustrated by the diagram on the next viewgraph…

Illustration of DTS



- Source(*src*) has *m* interfaces, destination(*dst*) has *n*
- Important issue: they are likely to be controlled by independent Internet Service Providers (ISP)s, each imposing its own firewalling

security is always an after-thought…

- We proposed novel solutions to the b/w aggregation problem (it is equivalent to the problem of striping data transfer across multiple IP links) in publications in *Infocom* (2002) ref [2], *Computer Networks Journal* (2003) ref [3], *IEEE JSAC*(2004) ref [4], ….

- b/w aggregation, jitter control, …. were the first attributes that were optimized in a DTS scenario

- Finally security**:** Leveraging multiple Interfaces for added Security (it is interesting that security almost always gets considered as an afterthought, which often leads to disastrous consequences ….)
  - better late than never ….. at least we considered security; as soon as b/w, jitter etc. were handled….

isn't a single link with strong SSI good enough ??

- A strong security infrastructure (SSI) includes
  1. Public key infrastructure (PKI) for key exchange, certification and authentication
  2. Strong symmetric-key algorithms for encryption (ex: AES) and verifying data integrity (ex: HMAC-SHA).
- Even a single link together with a strong SSI adequately addresses the basic security attributes, viz.

  Authentication, integrity, confidentiality and non-repudiation
- A strong SSI is already in place for critical applications (such as bank transactions, Real-time SCADA control systems, …)
- So what if anything can multiple interfaces add toward security ?

Must evaluate in the context of (single_link+SSI)

- There are two aspects:
  - Can multiple interfaces provide additional security ? Can they provide something that a single interface cannot ?
  - Can they achieve the same level of security (as a single link with a strong SSI) but at a lower cost in terms of computation overhead, latency, infrastructure required …. etc ?

intrinsic advantages of multiple links

- Even with an SSI in place, a single link is more susceptible to DOS attacks than multiple interfaces
- Multiple links provide higher
  - Flexibility
  - Reliability
  - Availability
- Higher throughput which can translate into security in critical situations
- It is possible to achieve same level of security at a lesser cost: due to the fundamental reason that the adversary needs to monitor all paths to mount {"Man-in-the-Middle" (M-i-M) and/or Replay} attacks

Equivalent security at lesser cost

Key agreement without public key cryptography

- Threshold cryptography is applicable: Split the key into $r \geq n$ parts (n = number of interfaces) and set the threshold $T = r$; transmit the r pieces along the n paths. All r pieces are needed to break the encryption
  - ∴ adversary must snoop on all n interfaces ⇒ more effort required for an attack to succeed
- The threshold secret sharing scheme can be extended to detect if any of the pieces have been modified, thereby making it feasible to detect the M-i-M attack

Mitigating M-i-M in unauthenticated DH−E

- M-i-M must be able to alter traffic, merely snooping does not defeat a "**Diffie-Hellman Exchange**" (**DH−E**)

  Multiple interfaces ⇒ M-i-M must be able to alter traffic on ALL paths which raises the effort level for an attacker

- If links are relatively safe, unauthenticated DH−E will work ⇒ Public key exchange which requires a PK infrastructure can be avoided ⇒ less infrastructure cost.

- Such a DH−E is a better method of establishing a shared secret than threshold cryptography alone, ∵ only snooping will suffice to break the code in the latter case

- Combination of the two. i.e., DH−E via Threshold Crypto is stronger.

Equivalent security at with less encryption effort

All-or-nothing Transform (AONT) : [Rivest 1997]

- Designed to increase security of Block Cipher codes: the entire cipher-text must be decrypted before a single message block can be isolated for crypto-analysis
- Rivest proposed an efficient implementation of AONT as a pre-processing step to be followed by regular block-cipher encryption
- Multiple interfaces make it possible to reduce the key length b_r of the block cipher encryption step, i.e., $b_r < b$
- In the best case, the block cipher coding can be skipped altogether.
- We make the independence assumption:

$$\text{Prob (breaking all } n \text{ links)} = P^n$$

Deriving reduced key length

- Consequently

$$\text{Expected \# of attempts to break all links} = (1/P^n)$$

- Average effort needed to break a b bit key is $2^{(b\text{-}1)}$
- Let E_{it} = effort to invert (decrypt) AONT.
- Equate effort needed to break in both cases:

$$\frac{1}{P} + E_{it} + 2^{(b-1)} = \frac{1}{P^n} + E_{it} + 2^{(b_r-1)}$$

$$\Rightarrow \quad b_r = \lg\left[\frac{1}{P} + 2^{(b-1)} - \frac{1}{P^n}\right] + 1$$

Block cipher coding (2nd step) can be skipped if the expression in square brackets ≤ 1
(for more details pl. see ref [1])

Need amplification : soln → SI

- For $b = 128$ bits, and $n = 4$, probability $P \lesssim 10^{(-10)}$
- Practical links are more susceptible ⇒ $n \gg 4$ is required to achieve a sufficiently low probability of breaking by brute force methods
- But the number of links "*n*" is physically limited, it cannot be increased or "amplified" to arbitrary levels

- Solution: "Spread Identity" over multiple IP addresses (via extended multi-homing) to amplify the number of interfaces ***apparent*** to an adversary in external network

SI, the first version (in Securecomm 2005) :
end-to-end and clumsy

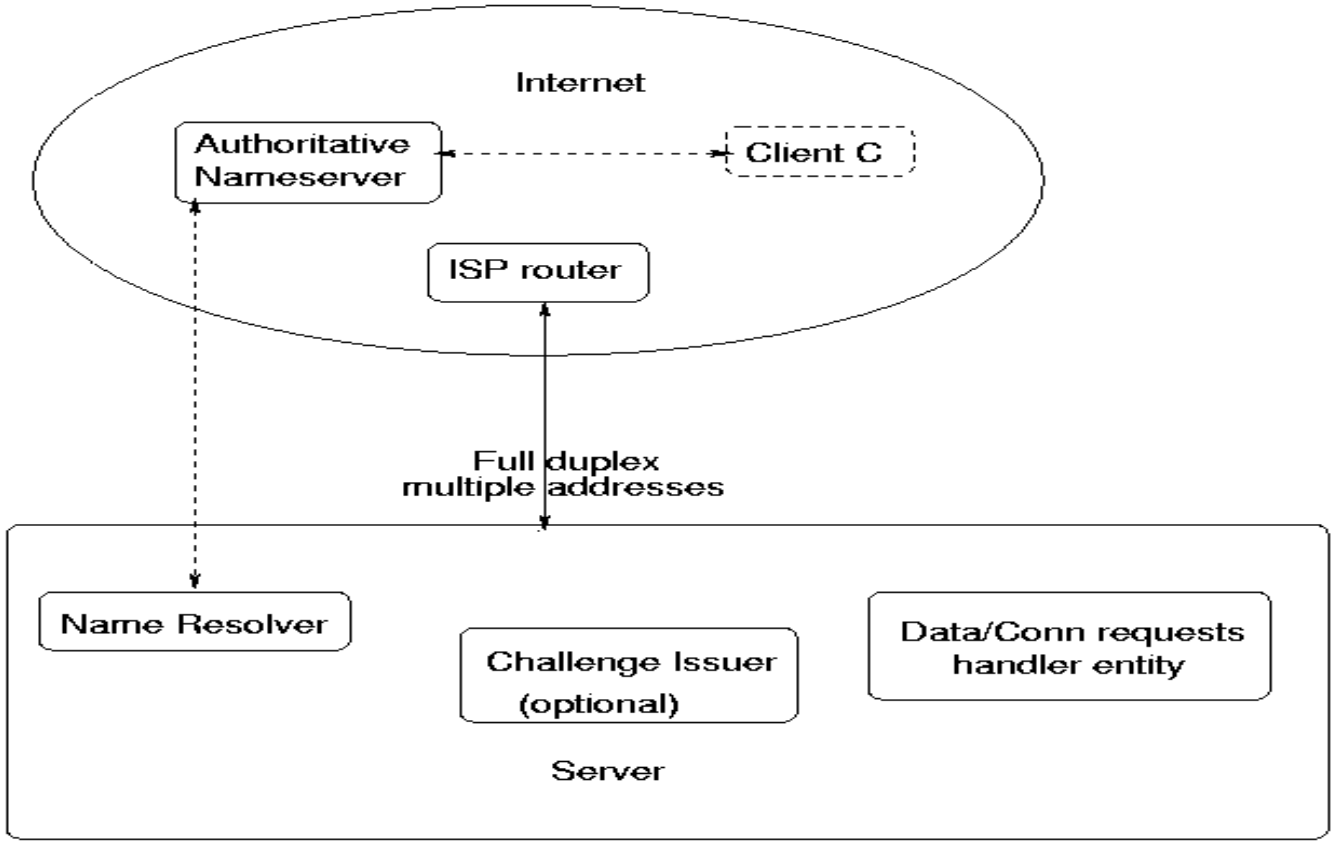
SI : Enabling mechanisms

- Dynamic Extended Multi Homing :
  - Dynamically assign and de-assign multiple addresses to any number of interfaces
  - For DDOS resilience, even a single interface with multiple addresses is sufficient
  - In effect this is tantamount to the ability to acquire multiple identities
- Control over name resolution (how the identity is spread)
  - Relatively easy to achieve: register host(s) under self jurisdiction as the authoritative nameserver with the IANA
- Address resolution strategies

Main idea : SI enables quenching of addresses

- Suppose address resolution works as follows: All queries for *D* that originate in the US get returned IP addresses from subset_1, all queries from China get addrs from subset_2 and those from Brazil get subset_3 addresses
- Now in case of a DDOS attack host *D* can selectively turn of traffic flows from say Brazil by doing two things
  - Tell the penultimate router (next-hop-router upstream) to temporarily set the routes to addresses in subset_3 to NULL; thereby temporarily "killing" those (self) addresses
  - "quenching" = a deliberate temporary killing of self IP addresses
  - Tell DNS server to return a null address if the algorithm lands upon an address which is being "quenched"
  - The routes can be restored later
  - The quenching is on destination address not source addresses so you control your own destiny ….

Server Architecture (old, can fast forward …)



DDOS resilient server architecture

Conceptually the server entity includes the name-resolver entity, the data/connection req handler entity and an optional challenge issuer. The entities could be processes within a host or nodes within an organization's premises connected by a LAN

Communication Protocol (older version)

1. Client C sends a req to resolve S
2. The authoritative Nameserver for S sends the query to entity S (if S indicates the capability and desire to do self name resolution say by setting a flag).
3. S doles out one of it's addresses say S_i
4. This creates a "token" (C,Si)
5. Future communications from client C must be addressed to "Si"

Leverage the name resolution process as an implicit token granting mechanism

The implicit tokens in turn enable multi level multi pronged defense against DOS

Problems with the first version

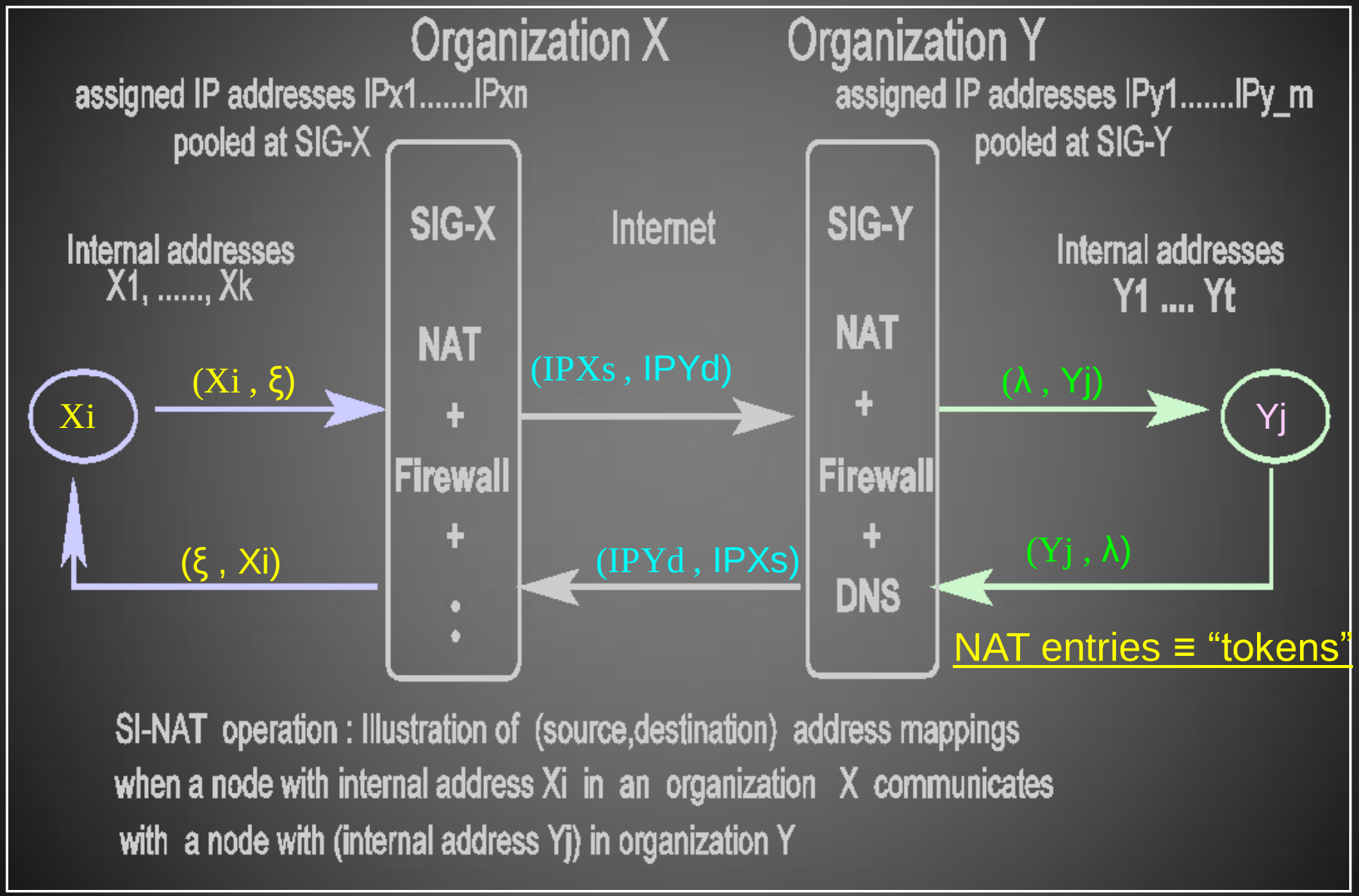
- # of IPv4 addresses available to individual hosts (to spread their identity) is very small, (IPv4 addresses are scarce…)
- Moreover, it is difficult for individual hosts to acquire, release, and "quench" IP address sufficiently dynamically
- Common practice today : Gateway at the edge performs NAT to hide the internal structure of the network
  - in all outgoing packets, the *src* address is replaced by one single IP address ⇒ all traffic from the *org* appears to come from one *src*
  - we call it a "single NAT" (∵ only 1 out of 2 addresses in the IP packet headers is NATed)
- NAT↔ Indirection, however, the "single NAT" does not exploit full potential of NAT/Indirection
- How to harness the full power of NAT/Indirection ?

The Edge-to-Edge transformation magically fixes everything : Silver bullet solution to multiple problems !!!

Address pooling and statistical multiplexing

- routable IP addresses assigned to an *org* ≡ "real addresses"
- "Pool" real addresses at the perimeter SI gateway and dynamically assign them at any desired level of granularity (such as {session, connections, pkts…})
- achieved via a double-NAT at the SI gateways at both ends
  - double-NAT at both *src* and *dst* ends fully exploits all benefits of "Indirection"
  - Tantamount to an indirection/virtualization from the end host's perspective (we call it "blindfolding")
- Each new session must be preceded by a DNS query
  - DNS reply creates a NAT entry in the Perimeter SIG at the *dst*
  - Upon receipt of the DNS reply, and checking that it is error-free, the SIG at *src* end also creates a corresponding NAT entry
  - NAT entries expire if no traffic is observed for a certain period.

double-NAT at both *src* and *dst* SI-gateways



communication protocol steps

- Source : host in organization (autonomous domain) X with internal address Xi (for example assume Xi is my machine photon.cs.umbc.edu)
- Destination : host in organization Y with internal address Yj (assume Yj is the default CISE web server "www.cise.nsf.gov")

- *Step 1* **:** Xi sends a gethostbyname query (example: Xi says "resolve www.cise.nsf.gov"
  - This query goes to the "internal" namesrver within the *src org* domain "cs.umbc.edu"
  - These resolvers are listed in /etc/resolv.conf files in unix (or their equivalents in other OSes) on individual hosts.
  - The internal server sends the query to the Authoritative DNS (ADNS) for the *dst org*

SI-NAT entry creation

- *Step 2* **:** in conjunction with the *src* name server, the *src* side SI gateway (sSIG) replaces the *src* address Xi with a pooled IP address, say IPXs (could be picked at random) and creates the temporary NAT entry

  (Xi, www.cise.nsf.gov) ⟹ (IPXs, www.cise.nsf.gov)

- the query reaches the authoritative DNS (ADNS) for the *dst org* Y (ex: NSF). The ADNS returns one of the pooled addresses: IPYd, and instructs the *dst* SI Gateway (dSIG) to create two mappings :

  (IPXs , IPYd) ⟹ (Λ , Yj) for incoming pkts and
  (IPYd , IPXs) ⟸ (Yj , Λ) for outgoing pkts.

SI-NAT communication establishment

- *Step 3* : ADNS reply (to IPXs) is received by SIG-X; where it is checked for errors. If there is no error in the DNS reply, then SIG-X replaces "www.cise.nsf.gov" with IPYd and creates the NAT entries

  (IPYd ,IPXs) $\Longrightarrow$ (Xi, ξ) for incoming pkts and
  (IPXs, IPYd) $\Longleftarrow$ (ξ, Xi) for outgoing.

  and relays the blindfolding addr ξ as the resolved address

- *Step 4* : From here on

  Xi communicates with virtual address ξ and

  Yj communicates with virtual address Λ

- *Step 5* : NAT entries are removed after seeing a FIN packet on the pair/token; or after a timeout

Blindfolding ≡ Indirection

- Internal hosts (ex: Xi) see only those IP addresses (ξ) that the SI Gateway in *org* X (SIG-X) chooses to show/reveal
- Internal hosts (ex: Yj) see only those IP addresses (Λ) that the SI Gateway in *org* Y (SIG-Y) chooses to show/reveal
- The internal hosts only see a virtual or "blindfolded" view of the external world, hence the name "blindfolding"

Source-side blindfolding : necessity and advantages

1. Necessary to solve a crucial problem: application caching
   - Applications "remember (i.e. cache)" recently resolved addresses
   - Ex: in the figure, suppose that an application on host Xi cashes the virtual address ξ for some time interval τ
   - SIG-X keep showing host Xi, the same address ξ for a time interval τ even if the *dst* end changes the real address IPYd
2. Allows use of arbitrary addresses in the internal network
   - Ex: Addresses assigned to Google could be used inside UMBC.
   - Q: How can UMBC hosts connect go REAL Google?
   - A: Using blindfolding: as long as the "blindfolding" address ξ used in the NAT entries is not one of the addresses used on the internal network, all traffic addressed to ξ will get routed to the perimeter SI gateway where the ξ is replaced by Google's real address so the packet does reach REAL Google.

Src side blindfolding advantages cont…

3. Enhances anonymity against a global adversary

   Global adversary = an entity that can snoop on either side of the SIG. (i.e., snoop on the internal network belonging to the *src org*, as well as the external Internet at large)

   ex: in fig. 1, a global adversary can snoop and learn that host Xi is communicating with ξ and IPXs with IPYd

   - if the SIG handles a lot of simultaneous flows, then further inferring that ξ is mapped to IPXs and vice versa is not that easy, esp if IPXs is changed frequently (CROWDS type anonymity that forces the adversary to do correlation analysis)
   - Without the blindfolding-address ξ a global adversary could learn by snooping on the internal network, that Xi communicates with IPXs; and (by snooping on the external net) that IPXs talks to IPYd. i.e., if blindfolding is not used on the *src* side, then merely snooping lets a global attacker trace a connection thru SIG

dst blindfolding advantages

- Quiz: What are the advantages of destination-side blindfolding?

Statistical Address multiplexing

- One single real IP address can support multiple concurrent inbound connections from distinct sources to same or distinct *dst* hosts withn the *dst org*; see the Example below

1) First : suppose that in response to a query from host_1.cs.umbc.edu for "cise.nsf.gov" the SIG/ADNS at NSF returns the real address, say "1.2.3.4"
2) Next : in response to another query from any_host.math.ucsb.edu for "math.nsf.gov" a different directorate, the SI-gateway/ADNS at NSF returns the SAME real address "1.2.3.4", it uses the source address to distinguish connections via NAT entries
   [IPXs, 1.2.3.4] $\Longleftrightarrow$ [ξ, Yj ∈ CISE] and
   [IPZt , 1.2.3.4] $\Longleftrightarrow$ [Λ, Yk ∈ MATH]

identity-to-ip_addr mapping can be many-to-many

Continuing the previous example…

3) Finally : suppose that another host within "cs.umbc.edu" (ex: host_2.cs.umbc.edu) wants to access the same destination, viz., "cise.nsf.gov", and suppose that the SIG/ADNS at NSF returns the real address "5.6.7.8"

- now, as illustrated in the diagram below, to an observer external to the *dst org* (NSF); the mapping between host identities and IP addresses appears to be many-to-many

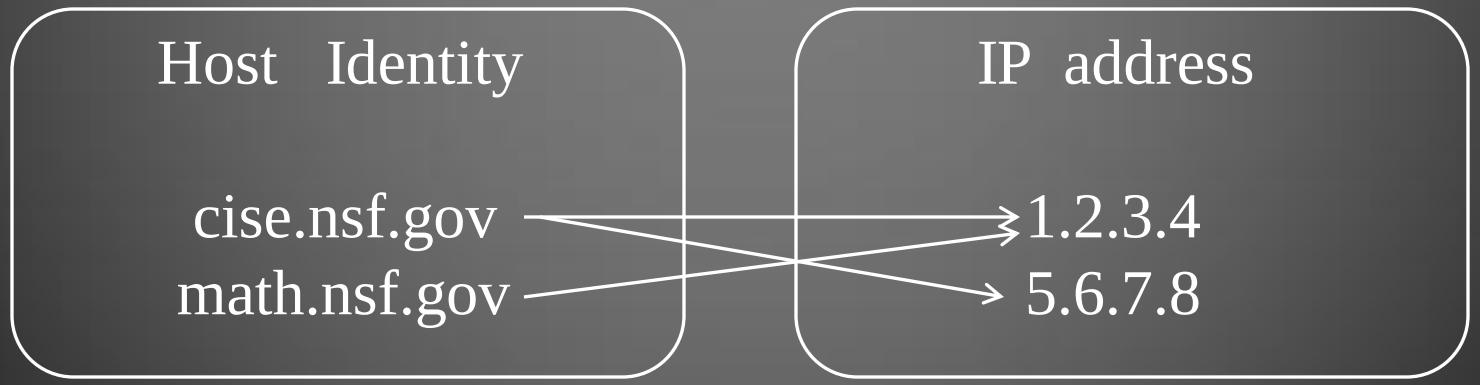


Illustration of many-to-many mapping

One of the main new ideas :

Leverage the SI-NAT entries as dynamic flow marker/tag => as dynamic access control tokens

1. Detect abnormal behavior extremely fast by simple IP level (src,dst) pair matchings
2. Mount a Multi level multi pronged highly effective DDOS defense (even aggressive defense is possible : re-direct bots against each other
3. Enormously simplify traceback : simply logging the SI-NAT entries captures connection history without the need to touch packet content
4. Substantially simplify control plane
5. Destination address itself serves as a flow "marker"
6. No access or egress past SI-G without a token => overall security is higher

Unique Major benefit 1 : Address scarcity problem is completely resolved

this benefit is incidental ; in other words SI was not designed to yield this benefit but ended up doing so anyway !

Address scarcity is gone …..

- The same works for outgoing connections as well: if a host *Xs* wants to talk to "google.com" and another host *Xt* wants to talk to "yahoo.com" then the fact that the destination addresses are distinct enables the reuse of the same real address as the source address for both Ys and Yt with the following NAT entries say it all:

  [Ys, Θ] ⟺ [1.2.3.4, IPG_m ∈ Google] and

  [Yt, δ] ⟺ [1.2.3.4, IPH_n ∈ Yahoo]

  - The same 1.2.3.4 could be being used for the cise web server …..

- External addresses can be reused on the internal network.
  - Extreme example : say goole's addresses are reused inside NSF
    The only question is what if host Yj wants to talk to another host Yk (both of whom have google's addresses) and also wants to talk to the real google:
- The solution is blindfolding : the SI-gateway intercepts Google's DNS response and shows a virtual address ξ for Google…..

of simultaneous connections can be unlimited

- The net effect is to be able to support a lot more connections without adding new fields or changing any syntax or the core routers.
- Note that in principle the number of simultaneous connections supported could be unlimited depending on the reuse granularity because the same pair of IP real routable addresses could be supporting connections in the US, Europe, Asia ……. Or at a finer granularity
- Addresses start to loosely resemble frequencies of a CDMA cell. The same frequencies can be reused in non-adjacent cells ……

Practical + commercially viable

Currently most organizations hide their internal structure anyway. That puts to rest the first question:

Q: How many entries are in the NAT tables ?

A: No bigger than NAT tables today. Fundamentally the number of connections has not changed.

Q: Creation/deletion ?

A: SI-NAT entries are self clocking : created upon a the egress of a DNS query and deleted after a FIN packet or after a timeout. period

$$$

- Advantages to the ISPs are obvious :
- Support more customers without requiring more IP addresses
- Charge a small fee for participation in an address pool: (could be marketed as “ we log all attempts to tamper with your home box: we protect your kids from Internet preditors). Subscribers have advantages also, explained later
- ISPs can sell lot of “add on” services: DDOS protection, providing packet sinks, traceback when requested the possibilites are almost endless we keep revisiting this issue throughout
- SI-Gateway implementation = $$ for router makers

This benefit is “incidental”

- The main reason we introduced address pooling was to make a large number of addresses available to dynamically spread the Identity.
- That works wonders for abnormal behavior detection, DDOS resilience and traceback.

Unique Major benefit 2 : Multi level multi pronged, robust defenses and even offenses to effectively mitigate all types of Distributed Denial of Service (DDoS) Attacks

DDoS Attacks

- In a typical DDoS attack, one attack bot gets an address for the target victim server S, distributes it to many (say thousands) of other machines (attack bots) and at a chosen time they all send requests to overwhelm the server.

- The goals are
- (1) To block genuine users from accessing the server
- (2) To the extent possible overload server memory, CPU and cripple it or crash it
- (3) If none of the above works, clog the input or output **bandwidth** (abbreviated **b/w**) by sending or requesting large volume of traffic or data, respectively.
- Accordingly, we first classify the DDOS based on what they target and show robust defenses against all of them

Types of DDoS attacks

- Type I : Resource consumption attacks
  - tie up memory/socket buffers (ex : TCP SYN floods, teardrop)
  - CPU overload (ex : by repeatedly invoking CGI scripts)
  - exploit asymmetry between client and server to clog **Output b/w**
- Type II : crowd out genuine users
  - Bots get tokens, behave normally and crowd out genuine users.
- Type III : Input b/w clogging
  - symmetric effort on part of the adversary
- Type IV : Attacks against DNS server/firewall
- Type V : Strategic all out attacks against entire pools or ranges of addresses

Tokens @ dst SIG = ultrafast abnormality trigger

Even if SI is implemented only at the server/destination side, It enables extremely fast detection of abnormal behavior due to the following causality chain:

1. traffic does not get past dst SI-Gateway and reach the server without an SI-NAT entry at the destination SIG
2. No SI-NAT entry without a name resolution query
3. So all the bots that got the address without a DNS query will have no “tokens” and their packets can be filtered off at the IP level

Furthermore The tokens enable Ultrafast detection of abnormal behavior

Token matching (at the IP level) enables Extremely Fast Detection of Abnormal Behavior

Normal client behavior	Abnormal behaviors rendered detectable by simple IP level token matching
Step 1: Xi sends a name–resolution reques (src addr=IPXs) It gets an address IPYd (one of the many Y has) /* This creates a pair/token (IPXs , IPYd)	(a) Casual query: only name resolution request, no data/connection requests detection method : token expires without aconn request
Step 2: Xi sends a data/connection request to address IPYd ○ ○ ○	(b) Unsolicited query: data request which is not preceded by name–resolution query detection : Source address does not appearin any token (i.e., in any SI–NAT entry)
Possibly more data requests from Xi to IPYd	(c) source sends data req to another address IPYk that Y has detection : Dest addr mismatch

- Works even if deployed only at the destination side
- Action by bots (flurries of DNS queries or token mismatched traffic) can be easily detected

First line of Defense: Rate limit name-resolution responses

- To get past the token matching defense, each bot must make a DNS query, i.e., to look like genuine client they must subject themselves to a DNS query
- => A fundamental defense strategy is to rate-limit the number of DNS responses
- Ex: if server capacity <1000 connections/second simply rate-limit # DNS responses to < 1000 /second.
- Excess requests are either delayed or returned an error code
- This way the server can limit the number of clients that can connect to it
- Note that this handles type I attacks: The server rate limits the number of connections so that it can survive rather than crash

Challenge-response test => genuine clients are not crowed out

- But if genuine users are crowded out, attacker has won ?
- There is a solution to that problem as well.
- First : when under attack (as evidenced by abnormally high DNS query volume) , the server re-directs all the clients to a challenge-response server to screen for humans in the loop
  - Genuine usage of the server is likely to be in the form of a person downloading/browsing/interacting ; a simple challenge response test such as rotated text will screen humans from automated bots
- Second : All sources that can be identified as bots get added to the (Bloom) Filter banks so the next time they send a query to the DNS server, that query is simply dropped.
- More proactive strategies outlined a bit later

Extremely fast detection and prevention of type II attacks

Rate limiting DNS responses + challenge-response test effectively neutralizes the worst type of attacks (where the bots try to masquerade as genuine clients by appearing to behave normally, and getting their tokens)

To circumvent the rate-limits and challenge-response tests the bots might try to send special types of floods (ex : TCP SYN floods, repeated CGI script invocations, etc)

(the conventional attack where one bot learns the address and distributes it to others)

Extremely fast detection and prevention of type II attacks continued

Note that all such traffic will “token mismatch” => can be filtered off at the IP level

- It never reaches the TCP or higher levels of the stack and therefore cannot consume resources

=> Ultrafast detection and neutralization of type II attacks

III attacks : Shrink identity = quench addresses

- Next possibility is INPUT b/w clogging attacks
- These are handled by selectively quenching addresses.
- In the previous example all addresses exposed to say Brazil could be “quenched”.
- To quench an address, the SI gateway does two things

1. Tells next hop upstream router to set routes corresponding to “quenched addresses to NULL (routes can be resurrected later)
2. The quenched addresses are not doled out in response to DNS queries (until they are re-surrected).

- Note : b/w clogging attacks are more symmetric: the attackers need to generate a large volume. The typical asymmetry between upload and download speeds further disadvantages the attackers
  + full duplex link to the ISP => output b/w uncluttered

Attacks against DNS server are easy to defend

- DNS server itself sits behind a firewall.
  - The (Bloom) filter banks can simply filter off requests from bots, they can never reach the DNS server
- A set of addresses/names can be reserved to spread the Identity of the DNS server itself (for example a simple "whois" query reveals that Yahoo has at least 5 name servers)
- So input-link bandwidth clogging attacks can be handled by "quenching" nameserver identities (for example, in the above case Yahoo could temporarily have the penultimate router set the routes to say {ns2,ns4}.yahoo.com to NULL and leave the remaining 3 address functional
- DNS query and response sizes are about the same => attacker cannot exploit asymmetry ; the attacker must generate lot of traffic

Last gasp

- The only thing now left is an attack on the SI-Gateway itself or a penultimate router.
- Since the SI-Gateway is controlled by the ISP, in effect the adversary now has to bring down a full fledged ISP router/gateway
- This is raising the adversary's effort level substantially

- We argue this is success ! The Attacker cannot touch anything inside the SI-Gateway/firewall, All that is left to do is to try and bring down SI-G or penultimate router.

- Even this can be foiled

Re-direct bots against one another

- The most effective attacks are of types I and II. To be able to mount these attacks, all bots must have a “token”
- => they must make name-resolution queries
- If based on past history and by other means, if S can determine that a “flurry” of name-resolution queries is from an army of attack bots then
- S can return them each-other’s addresses: let them clobber each other !

Snuffing attack flows right at the source

- ***Reflection*** **:** Return a bot it's own address in respone to a DNS query

- Now the traffic never leaves the bot node (the script kitties will soon realize the "reflection offense" and will modify the scripts to not generate traffic if destination addr is self addr. This is win for all: a potential attack flows are prevented so network resources are not consumed at all.

- Even if the attackers eventually figure out all they can do is send another query or try to clog the bandwidth

Honeypot "trickle" TCP + apps

DNS response returns the address of special host that has

Specialized TCP stack and applications to

1. Keep the bot connected
2. but send it the bare minimum trickle so it does not drop connection
3. This way the bots think they've brought the server down
4. Most important: keep as many bots "connected but idling" as possible.
5. For the attacker to figure out this strategy the bots will have to "collaborate" exchanging data about transfer rates and other parameters => now the attack management itself becomes more complex. That should thwart most script kitties

Alternatives to reflection

- For a fee ($$) the ISP could provide packet sinks in physically separate locations to trap bot-traffic for traceback/analysis ….
- Redirection to such a physically separate packet sink has the added benefit of unclogging the victim node's input bandwidth

Active baiting, identity traps

- Forcing the bots to follow an “identity trail”
- Example : default policy might be to give out a new dst address for each new query (from a different source address).
- When under attack, the SI-G starts giving out the same address thereby “herding” all attack bots in a “corner of the name space”. Those addresses can then be quenched.
- Whoever else token mismatches on such a “bait” value of destination address gives themselves away as a belonging to a cluster of bots
- => identifying and clustering bots is very fast. The bloom filter banks can the be armed to filter unwanted pkts.

Successive attacks are automatically downgraded

- Suppose “smart” bots get tokens and try to masquerade: the challenge response can screen them out.
- They are now entered into bloom filer banks
- Not only is the first attack neutralized, the next time the same bots try to send traffic it will be filtered off at the IP level (bloom filters are armed this time around)
- That downgrades them to b/w clogging type attack the next time they try
- Consecutive attacks from the same set of source addresses are progressively downgraded into lesser and lesser effective categories.
- This is nice: each attack (punch) strengthens defenses

Preventing future attacks by snuffing them right at the origin

- Inaction by bots can also indicate an attack in the making:
- For example, if a token expires without a query and the query was from a host that is in one of the bot-clusters then that cluster is likely to be active again so the Address revealing strategy can be preemptively set to do the following:
  - Return the source's own address if the source belongs to that cluster
  - Now the attacker cannot send traffic anywhere (name-resolution returns its own address
- Bottom line : Action by bots gives them away, INACTION gives away an potential attack in the making as well.

Better than the stock market

- stock market :

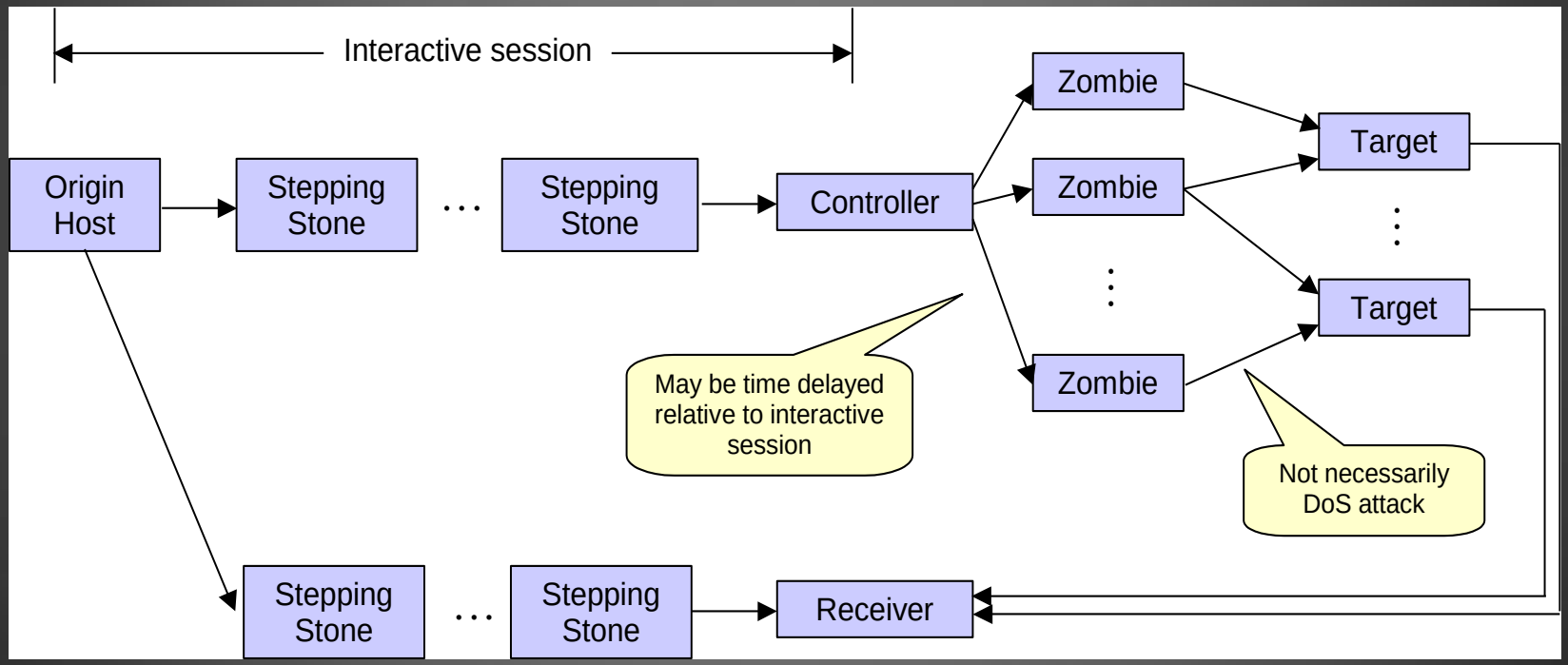
  win when prices rise ~ win when bots act
  win when prices fall (shorting) ~ INACTION also is a win

- In the stock market, the whole shorting phenomenon is dependent on the fact that eventually the prices will rise again.

- With Spread Identity we unconditionally win with ACTION and INACTION (i.e., there is no need for the inaction to be eventually followed by action-by-bots for the defender to become aware of the intention)

Dimension 2 : Traceback is pretty much solved

- The attack scenario (Source : BAA_04_05_04IFKA_NetworkTraceback



Current Methods and their drawbacks

- End host packet auditing :
  - If the end hosts are compromised, all logs can be deleted or corrupted
  - In the event of a traceback, ordinary hosts might not be equipped to mine the logs and answer the queries

- Network packet logging : brute-force: routers log all pkts
  - Scalability nightmare despite almost endless series of tweaks like compressed hash per pkt, log only a subset of pkts
  - We think that approach is fundamentally wrong because

1. Adversary can obfuscate (by using encryption, chaffing, inserting "randomized delays" in the chain of command). All pkt logging can retrieve in such cases is "connection history"
2. It goes against the future trend where more and more data will go through IPsec, ssh ssl, and other encrypted channels.

Tracback mechanism : desirable attributes

- We focus on the main attributes that a good traceback scheme should have
- It cannot depend on end hosts for logging/auditing
- Packet content analysis should be avoided since it can be defeated by obfuscation (encryption, chaffing, randomized delays …)
- Scalable (Routers should be left alone)
- There a solution that satisfies all these : SI-TRACEBACK

SI-NAT entries = record of communications

- No egress/ingress across SI-Gateway without a token => every packet delivered belongs to a communication whose peers are listed in the SI-NAT entries
- Simply log the SI-NAT entries before deletion
- This captures a record of actual as well as intended communications (who wanted to talk to whom at what time…)

Efficient, scaleble, simple, accurate, robust

- No need to touch individual packets or their payload
- Only the SI-gateways do logging => the logging is not dependent upon end hosts
- Intermediate Routers are left alone !
- As accurate as network packet logging : because when faced with obfuscation, all network pkt logging can retrieve is a record of connections
- We directly zoom in on the connection logging with minimal effort
- Logging is at both ends (like double-entry-book keeping) => inherently safer: cannot be subverted by either src or destination alone
- Flexible : “suspect connections” => log more info (beyond simply duping the SI-NAT entry to the logger)

Even if SI-NAT is deployed ONLY at the desination end, it still simplifies traceback a lot.

New traceback tool no. 1: src addr in DNS query

- Identity of the last hop in the chain of command must be revealed
- Since the attacker does not know which IP address will be returned, it must have the ability to READ the DNS response => the DNS query cannot be made with a spoofed source address : either the source address is the real address of the last bot or the attacker can “read” messages sent to that address
- Either way the last hop node in the DNS query path is necessarily exposed.

New-traceback-tool 2 : token expiry time

- Another benefit of SI even if it is deployed only at the destination end.
- Attackers use encryption, chaffing, random padding etc to thwart payload scrutiny at intermediate nodes.
- That leaves only a “timing” correlation analysis as the only diagnostic tool: if a pkt on link “i” is seen to precede a pkt on link “j” then they could possibly indicate the attack-path.
- Correlation analysis can be fooled by inserting random delays + fake traffic at each “stepping stone” node.
- Such random delays can be “squeezed out” by dynamically adapting the token-expiry-time

Novel traceback capability 2 continued

- Adapting includes both
  - relaxing (to tempt the attacker into sending the payload by providing a sufficiently long expiry period)
  - Tightening (to thwart insertion of delays at stepping stones)

Redirective Traceback : new traceback-tool 3

- Even if is SI implemented only at the destination end, it enables yet annother new mechanism: Proactive traceback
- Ex : target is the DoD web server, suspected attacker is in the US but apparent source addresses (spoofed) from say Europe.
- Now the DoD could reutrn the address of its sites in Europe and sensitize any sniffers it may have on the links between US and Europe to scan for packets to that specific destination address (that the DoD just doled out)
- In other words the ability to dynamically direct the attack traffic to other hosts (geographically or topologically distant) can help localize the actual source of the attack

SI enables Proactive traceback even when it is deployed at only the destination end

- Obviously, combining the 3 tools as required together with past forensic info can substantially improve traceback
- Note that even if a traceback episode fails, it is likely to generate more forensic data than the corresponding case where the destination identity is static (literally a "sitting duck"
- SI gives the sitting duck the ability to fly ;; forcing the attackers to chase it, thereby increasing the chance that the attackers will stumble across some tripwire ……..

SI at both src dest ends => traceback trivial

- If SI is implemented at both ends and the Source side SI gateway cooperates with the destination side SI gateway, then traceback is trivial....
- No access to the Internet (at the source end) unless a valid token (NAT entry) is created by the SI Gateway at the Source end
- => Simply keeping track of the tokens=NAT entries at src and dst. ends has all the info needed for traceback…

SI-NAT logs = abnormality detectors

- Discrepancy between src and destination logs can detect attempts to subvert the logs
- Periodic scanning of logs to identify tokens that expire without data exchange and cross-correlating such “casual queries” can discern larger bot clusters.
- Correlating the logs with token mismatched traffic….

SI-NAT logs worth their weight in $$$

- Beyond traceback and other more immediate uses, SI-NAT logs will be valuable :

- Advertizers will pay a high premium for accessing those logs : because the SI-Gateway is now a barrier beyond which the Advertizers cannot “see” clearly, so they’ll have to request access to those logs to “shrink the identites” of the entities visiting their (or their competitor’s !) sites.

- Throw in HEPA/medical privacy issues (who can control and access the logs … then the lawyers get into the act and we’ve a got a thriller better than the Superbowl…..

Archival, trend analysis, capacity planning….

- The energizer bunny is running again ….
- SI-NAT logs should have more lasting value beyond helping mitigate the short term fracas (DDOS attacks, traceback, greed driven usages of the logs ….)

- long-term archival (a succicint history of connections !)
- Capacity planning and architecture modifications

- Trend analysis : (more general than capacity planning) In an “information centric” world connection logs would be the foundation of everything else !!!!! ($$)^($$)

other benefits : simpler control plane

- Only src, dst SI gateways need to collaborate to filter/process flows, intermediate routers need not participate in it.
- Reflected or non-routable address in response to a DNS query could be overloaded to signal different levels of "throttling" to be implemented by the source-side SI-Gateway.
- Ex: dst SI-gateway SIG-Y, could signal SIG-X thus:
  - Resolved addr ≡ source addr => block that source temporarily
  - Returning 127.0.0.1 ≡ stronger choke signal: block all connections
  - Return-addr = 0.0.0.0 => Block connections + DNS queries
- Assumes communications between SI-gateways are and authenticated and tamper-proof (which is easy to do)

Overall security is improved

- Only src, dst SI gateways need to collaborate to filter/process flows, intermediate routers do not
- Destination IP address itself could be used as a “tag or marker”
- No communication without SI-tokens => stealthy port scans and other vulnerability probes can be filtered
- Compromising end hosts to use them as bots/stepping stones is now a lot less effective for multiple reasons

Destination address itself serves as flow “marker”

- Destination IP address itself could be used as a “tag or marker” which makes it easier for intermediate routers to filter traffic (large number of clients typically talk to a server => filtering on the destination address of the server is easier

- If SI-NAT pools are hierarchically deployed together with an aggressive address sharing/reuse, then core routers need to maintain info about fewer end systems => routing tables and BGP/RIP (topology update and maintenance is easier

Overall security is improved

- No communication without SI-tokens => stealthy port scans and other vulnerability probes can be filtered
- Compromising end hosts to use them as bots/stepping stones is now a lot less effective for multiple reasons
  1. Blindfolding => bots cannot attack a target directly by IP address, they must go through DNS query step => rate limits can always be enforced.
  2. If source-side SI-gateway is not compromised, it will enforce the choking/filtering the destination is requesting
  3. => the adversary now has to compromise SI-gateways which is likely to be harder (gateways are more likely to be monitored and are less accessible)

Other Potential benefits

- Privacy/anonymity is bolstered
  - Does not contradict "Enhanced-Traceback" claim:
  - End-to-End Host anonymity and Edge-to-Edge traceability are both enhanced simultaneously

- Yet another example of how Edge-to-Edge achieves the optimal compromise between conflicting attributes: Tracebility and Anonymity

Address Chaffing

- Src and Dst SI-gateways can further obfuscate: stick different IP addresses in different packets belonging to the same connection.
- Packets belonging to a connection are identified by a flow identifier.
- flow-id = E(actual-src || actual-dst || random-pad)
  - Random pad and encryption prevents replay attacks
  - The protocol is stateless (i.e., dst gateway need not keep track of any sequence numbers, retransmissions…..)
- Address-chaffed connections presents another value-added service for which the ISPs could charge a fee.

Implementing Address Chaffed connections

- Source Xi requests address-chaffed connection to Yj
- SI-Gateways exchange encrypted *asrc* and *adst* values
- From here-on, prepend flow-id to the payload:

Sender : pkt-data → E(*asrc* || *adst* || pad) || pkt-data

IP header : pick any src , dst ∈ the address pool

Receiver SIG : retrive (*asrc* , *adst*) and NATs accordingly

TCP flow control (Acks, retrans …) is still true end-to-end => no state info in SIG => scalability is automatic.

Address chaffing does not hamper EG-to-EG traceback: (*asrc* , *adst*) entry gets logged at both ends

Why the name "chaffing"

- Chaffing and Winnowing was introduced by Ron Rivest in 1997 mainly to circumvent the export control restrictions
- It is clear that obfuscating addresses as outlined above is essentially tantamount to a "chaffing and winnowing"
- Rivest's scheme chaffed data, we chaff connection identifiers

(This correspondence was identified after the fact, i.e., after the address obfuscation was devised)

Spread Identity and PKI

- SI, address chaffing and the other mechanisms presented herein make M-I-M (man-in-the-middle) attacks more difficult
- If M-I-M is not a threat will a Diffie Hellman exchange suffice to dynamically establish a shared secret ?
  - The only vulnerability of Diffie-Hellman is M-i-M
- The goal is to skip elaborate certificates and authentication for all but the most critical communications

Fundamental Issue : Efficient and flexible Name-space design

Must curb the urge to uniquely tag everything

IPv4 32 bits, Ethernet 48 bits, IPv6 128 bits….

- Why don't we go to 256 bits once and for all ?
- 2^256 >> number of molecules on planet ocean (mistakenly called earth)

- Every entity could have its own address range for its exclusive use thoughout the conceivable life of that entity and still the address space would not get exhausted.

- If IANA gave me such an exclusive address RANGE for my own use, I would refuse it: why ??

Because it is the exact anti-thesis of anonymity

- I would rather deliberately share a (relatively small) pool of IP addresses with a lot of peers for crowds like anonimity ……
- Fundamental issue : Translation of Name-Spaces.
- Bring in the best practices from architecture (virtually addressed and physically tagged caches), compilers ……

efficient pkt routing = THE most important function of a network and its protocols

- No matter how long the addresses are (ex, say 256 bits)
- Note that the addresses must be allocated hierarchically in a Topologically meaningful manner
  - Otherwise routing tables (esp. in core routers) would explode (in the worst case needing 2^(256) entries in theory ; if the addresses were allocated completely randomly)
- In other words making routing fast and efficient must be one of the main goals of any network/protocols design
  - To achieve this core functionality (viz., routing pkts fast and efficiently) necessarily implies hierarchical and topologically meaningful allocation of addresses which allows an efficient "globbing" of large number of individual addresses and routes.
  - This in turn implies that a sufficiently large number bits in the very long address must be same/invariant across topologically neighboring sub-networks

long addresses by themselves do not avoid static assignments

- These “invariant” parts of the address (typically most significant half) become the “static” network identifiers
  - for example : IPv6 host address is 128 bits long ; but the more significant half or 64 higher-order bits are the same for all hosts belonging to a sub-network
- => our goal of avoiding static “names/tags/handles” is defeated….
- In other words, simply making address very long does not avoid the problem of static assignments

long addresses do not increase anonymity either

- Worse yet, the static identifiers of sub-networks are bad for anonymity/privacy : it is easy to identify the topological sub network which includes any give host
- Thus we have demonstrated that making the network address long does not help pretty much anything
- Long addresses simply waste bandwidth without giving anything significant in return….

Addresses pooling and their dynamic allocation to achieve mappings that appear many-to-many by deliberate design is the key idea !

anonymity guaranteed in hardware by design !

- The mapping between identities (of internal hosts/resources) and IP addresses is deliberately made to appear many-to-many to any observer on the either side of the SI gateway
- Only the gateway knows the underlying one-to-one mappings
- If the gateway does not reveal/output the NAT entries it is not possible to identify with certainty who is connected to whom (many-to-many mappings are not invertible)

Two SI gateways in series = MIXnet ?

- The SI gateways could be realized fully in h/w => unless the h/w is configured to output the NAT entries ; then traceback to identify who is talking to whom will be very hard if not impossible.

  Two such gateways in series is equivalent to the MIXnet primitive from cryptology ??

- If DNS queries and responses are encrypted and the spread identity is extensively deployed (so that even if someone finds out the resolved-address returned it is not that easy to find out where the actual destination is (because of the evesdropper tries to use that address, their pkts go nowhere since the token was not created in their name)

Crowds+TOR like anonymity in a h/w box

- Note that router manufacturers could incorporate SI gateway functionality at the edge-gateways and sell these boxes as providing anonymity guaranteed by h/w
- same benefits as CROWDS/TOR without the drawbacks of either (such as long delays ….)
- This could be valuable to customers whose business model values anonymity (ex. the Swiss bank)
- The h/w could be designed to archive or output its NAT entries if sufficient proof of authority (such as a warrant authorizing wiretaps) is furnished ….

Making "Big brothering" impossble

- If these mechanisms were in place or were to be phased in quietly, then simple "stifling" of contact with outside world (ex. by Iran , China , …., etc) would be rendered difficult if not impossible
- Let freedom ring true for all !
- Arguably the terrorists also benefit (their connections also enjoy the same anonymity etc)…..
- We think that the benefits outweigh the risks

Connections to QoS

- DNS query and response “sets” the NAT entries
- If there are 2 or more gateways in series then this process of setting the NAT entries is equivalent to “setting the path” in a circuit switched network.
- If no (or very few) addresses are allocated to a flow it can be starved of bandwidth and/or router buffers etc.

- Addresses themselves start looking like “bandwidth”=> a direct connection to QoS

All this + benefits documented in prior literature

- Indirection (**) is a very powerful principle
- " ** = solution to all problems in CS " famous quote !!!

…. Ooops if indirect means delegete (re-direct ur work to others) and it is one of the fundamental problem solving tools in CS then what kind of a science is CS

It is the only science that is exception to the rule

"if it has to be called a science then it is not a science"

Indirection has been used time and again

- Micro-programmed control in 60s
- Virtual memory …..
- Indirection subsumes translation : Compilers translate our virtual/abstract constructs into machine language

- The examples are endless

- DNS is also an indirection where the result of the indirection (aka name-resolution) process is communicated to the end host (E-to-E)

Indirection schemes have been proposed … but

- Given that indirection is so useful and powerful, it is natural to consider its use in the Internet.
- This has been done, several indirection schemes have been proposed in the literature:
  - TRIAD [Cheriton, Gritter et. al],
  - Internet Indirection Infrastructure (III) [Stoica, Shenker et. al.]
  - The list is growing ….
- Indirection is shown to bolster host mobility, load balancing, throughput, reliability etc.
- Security is not adequately addressed (for instance III does consider DOS resilience but not in sufficient depth)
- None of the works published in the literature consider the deliberate spreading of identity for security

New contributions-1

- Our novel contributions include the following principles, mechanisms, and their integration into a simple clean and efficient architecture.

1. Spreading identity for
    1. Ultrafast abnormality/attack detection
    2. Highly effective Multi-level multi-pronged defenses against DDOS
    3. Network traceback
2. Address (pooling , sharing + dynamic resolution) to enable spread
3. Leveraging Indirection and other novel mechanisms (including the ones listed in item 2 above) in order to realize an effective spreading of Identity

Contributions-2

1. Seamless integration of new SI mechanisms + Indirection + other existing schemes into a simple clean and efficient architecture that has enormous advantages detailed throughout this document

Fundamental Theoretical issues: Contrib-3

Fundamental Theoretical contributions

- Spreading-Identity is also a fundamental principle (like the principle of Indirection and the End-to-End principle)
- End-to-End principle is often at odds with the principle of Indirection.
- If E-to-E and ** are not synergistic then EG-to-EG achieves the best compromise

Next generation Internet should carefully balance all 3

SI-the meta concept is universally powerful

- Internet is just one application of the SI principle

- The fundamental Information Theoretic principle

  Spread-Identity principle :

  Any time a set of Identifiers (names/numbers/objects/entities) is translated into another set of identifiers, the biding should be late and dynamic;

  Additionally, the mapping should appear as many-to-many to all entities except the one that is performing the translation.

(the underlined text in the light blue color is our contribution)

Examples

Simplest : when writing a paper the first tendency is to assign the first reference cited the number 1 and so on. That kind of a “static” translation is not good (that’s why latex needs to be run twice after running Bibtex)

Static/early bindings are bad

- Next example : Compilers generate virtual addresses that are dynamically bound to physical addresses at run time
- Credit-cards : Financial Identity tied to one number, bad in the long run
- Social Security : Identity at large tied to one object (a number) again a disaster in the long run
- Our sysadmin told me the other day that he cannot upgrade CS dept to Kerberose because the UIDs on master kerberose authenticator are in conflict with current CS assignments (same problem again)

Potential applications everywhere $$$$

- First do a good job with SI internet architecture
- Then take that experience into the credit card world
- Then try to tackle the Social Security issue

- Operating Systems : devise mechanisms to dynamically bind apparent UIDs to real UIDs that are stored internally and known only to the OS, not even to the root (like passwords, root can change a passwd it can decrypt)
- Apply the same concepts to Databases

- Wild dream : make bill gates pay : a back-end compiler for spread-sheets that virtualizes the fields so that if and when Wal Mart buys K mart their inventory databases/spread sheets can be merged easily

Getting carried away (the ultimate sales pitch?)

- Spread is intrinsic to everything, natural or human-made
- Portfolio is "spread"
- Sexual reproduction is evolutions attempt to "spread" the genetic pool to maximize the chance of the survival of the species
- At the most fundamental level: The Heisenberg Uncertainty principle can be thought to demonstrates the "spread" into the very fabric of space-time-mass continuum we inhabit
- Transcending the physical realm, even mathematics the purest of pure collection of immutable truths includes notions like information theoretic entropy

- ∵ Spread is fundamental and all pervading, why don't you license my SI patent(s)??

Epilogue : I practice what I preach

- Currently, my "eigen-values" are "SPREAD" across at least 3 distinct areas (In other words I am currently interested in the following 3 research areas):

1) Computer Arithmetic Algorithms and their hardware (VLSI/FPGA …) realizations

2) Cryptology, number theory, h/w implementations of crypto algorithms

3) Computing and Information Security (all aspects, including software/hardware/system/network security, information assurance, reliability, malware
…… you name it !!!)

Evolution of hot and trendy areas

- Early to mid 90s everyone wanted to be an expert in computer--networks.
- Next band-wagon was mobile IP,
- then the fad was wireless
- That lead to ad-hoc networks.
- The next incarnation is/was sensor networks
- Today : everybody and their brother wants to re invent themselves as a Cyber xxxxxx GURU
  - xxxxxx = anything ; preferably Security

Current Area-1

- Computer Arithmetic Algorithms and their VLSI realizations
- This is a hobby that excites my eigenvalues.
- Have done substantial work
  - Redundant Sum Addressed Cache
  - Double Step Branching CORDIC algorithm for fast Sine and Cosine generation (US patent #)
  - Redundant number representations
  - fast straight remaindering algorithm for very large word-lengths (ARITH-17)
  - Fast modular exponentiation
  - Residue Number Systems

past work

- Neural Networks

- Phd thesis (strong negative results debunking the “folk theorem” of intrinsic fault-tolerance of ANNs), good pubs,

- National Science Foundation’s (NSF) CAREER award.

Past life = work in distant past

- For my MS thesis : I solved Maxwell's equations for a living …..

Thank you for the time and consideration

Questions? (contact phatak@umbc.edu)