Simulator

Krishna Deepak

ETH Zurich

Contents of the Talk

SCION Simulator

- Simulates functionalities of SCION elements (BS, CS, PS, ER)
  - Beacon Propagation at BS, Packet Forwarding at Routers, IFID propagation etc
- Implemented as a Discrete Event Simulator
  - Functionalities of the SCION elements are modeled as events
  - Each event is associated with a timestamp
  - Events are scheduled using a priority queue

Implementation Challenges

- Single-threaded vs Multi-threaded
  - Simulator is implemented using a single-thread
  - Actual SCION implementation is multi-threaded
  - Simplified implementation
    - No redundancy of servers (i.e., single beacon server + eliminated zookeeper)
- Need to map recv() calls of scion elements to a special - sim_recv() function as we do not have any sockets involved

Implementation Challenges

- Functions which are to be called repeatedly(e.g., beacon propagation) need to be called recursively so as to ‘schedule’ the next event at a later time
- Many crypto operations are removed from simulator as they were the causes of bottlenecks
  - Verifying MAC in OF’s
  - Verifying beacons and TRC’s

Implementation Challenges

- All the infrastructure elements have a simulator version of them in which we override any function to be modified
- Real time vs Virtual time
  - Simulator runs on a virtual time scale
  - Hence, all the timestamps(time.time() calls) in the actual SCION code are replaced with simulator's current time(virtual time)

What can be done with simulator?

- Code Debugging
  - Can schedule specific events and see if code runs the way we want it to
  - Ideal for testing corner cases

- Functionality Verification at Large scale
  - Run the codebase on large topologies
  - Use Topology generator to generate large topologies

Issues with simulator

- Changes to SCION Codebase requires manual change in the Simulator

- Inherent Scalability Limitations
    - E.g. A ping pong application on a topology of 6 ISD’s and approximately 500 nodes takes almost 70 sec to complete

SCION Topology Generator

BRITE

- BRITE is an Internet topology generator
- It can be used for generating flat AS-level topology, flat router-level topology or a mixed one
- Many internet models such as Barabasi-Albert Model, Waxman model are incorporated in it so that the topology is indeed Internet-like
- Although BRITE is no longer supported, it is quite good for our purposes

BRITE to SCION

- Convert each Brite file into a different ISD
- Challenges
  - No notion of ISD’s
  - No business relationship among the edges
  - Coming up with ways to interconnect core AD’s in the ISD’s

BRITE to SCION

- Steps in the conversion
  - Each brite file specified is converted to an ISD
  - Core AD’s are identified and all edges are labelled into one of the four options - ‘Routing’, ‘Peer’, ‘Parent’, ‘Child’
  - The ISD core’s are then interconnected manually using min and max degree(can be specified at command line)

Generating an ISD

- From the given brite file, some AD’s with high degree are chosen to be the core AD’s for an ISD
- Breadth-first Search is performed starting from these core AD’s to mark Parent-Child relations among the edges
- Any edge between two nodes which are on the same level(while doing BFS) is considered a Peer edge
- Then, we interconnect ISD’s using the min and max degrees specified

Interconnecting ISDs

- We generate a model ISD graph in which we consider each ISD to be a node. Using this model graph, we will generate the actual inter-ISD routing edges later
- Steps involved
  - Connect all ISD’s in a cyclic fashion to ensure that they remain connected
  - Using min degree parameter, add edges so that all nodes have at least this degree

Continued..

- Steps continued
  - Now, edges are added only between a chosen set of ISD’s to make them denser and Internet-like
  - Edges are added randomly between these chosen ISD’s, ensuring the max degree specified is not exceeded for any ISD
- Using the model graph generated, routing edges are added between randomly chosen core AD’s from the two ISD’s to be connected to generate the final graph

Command line options

- The min and max degree of interconnections can be adjusted using command line options
- Command line options are also available to convert all files in a specified directory (or) to specify each file to be converted separately

An attack on SCION

The problem

- MAC size in an Opaque Field(OF) can be changed by the AD
- In the worst case, we assume that all AD's set this to 24 bits - 3 bytes
- Attacker can juggle with the opaque field and perform a brute force attack with all possible MAC's
- Using this fact, various attacks might be possible depending on the goal of the attack

Goal of the attack

- Attacker wants to change the timestamp of a down-path so that he can use it for a longer period
- Timestamp field - TS
  - Present in Special Opaque field(SOF) which is different for an up-path and down path
  - The first router on the down path updates the EXP Time field in SCION packet header with the down path timestamp from SOF
- The attacker needs to change the SOF of down-path

Preliminaries

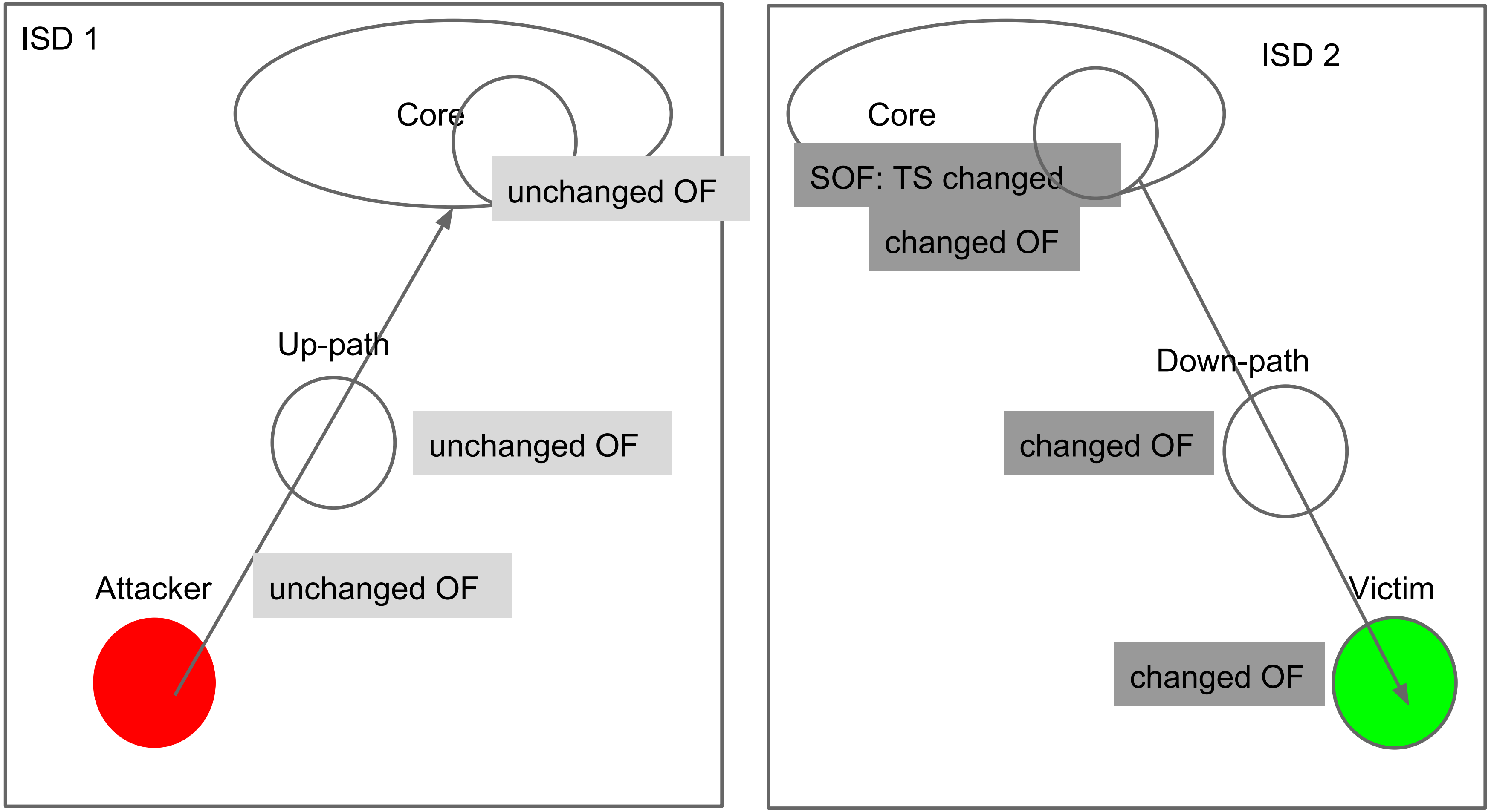
- Each edge router in an AD computes the MAC key using TS and a key local to the AD

MAC key (or) $K_{MAC} = F(TS, K_{AD})$

- Using this key, MAC field is computed in the following way

MAC field = MAC_{KMAC}(ingress, egress, prev. Opaque field)

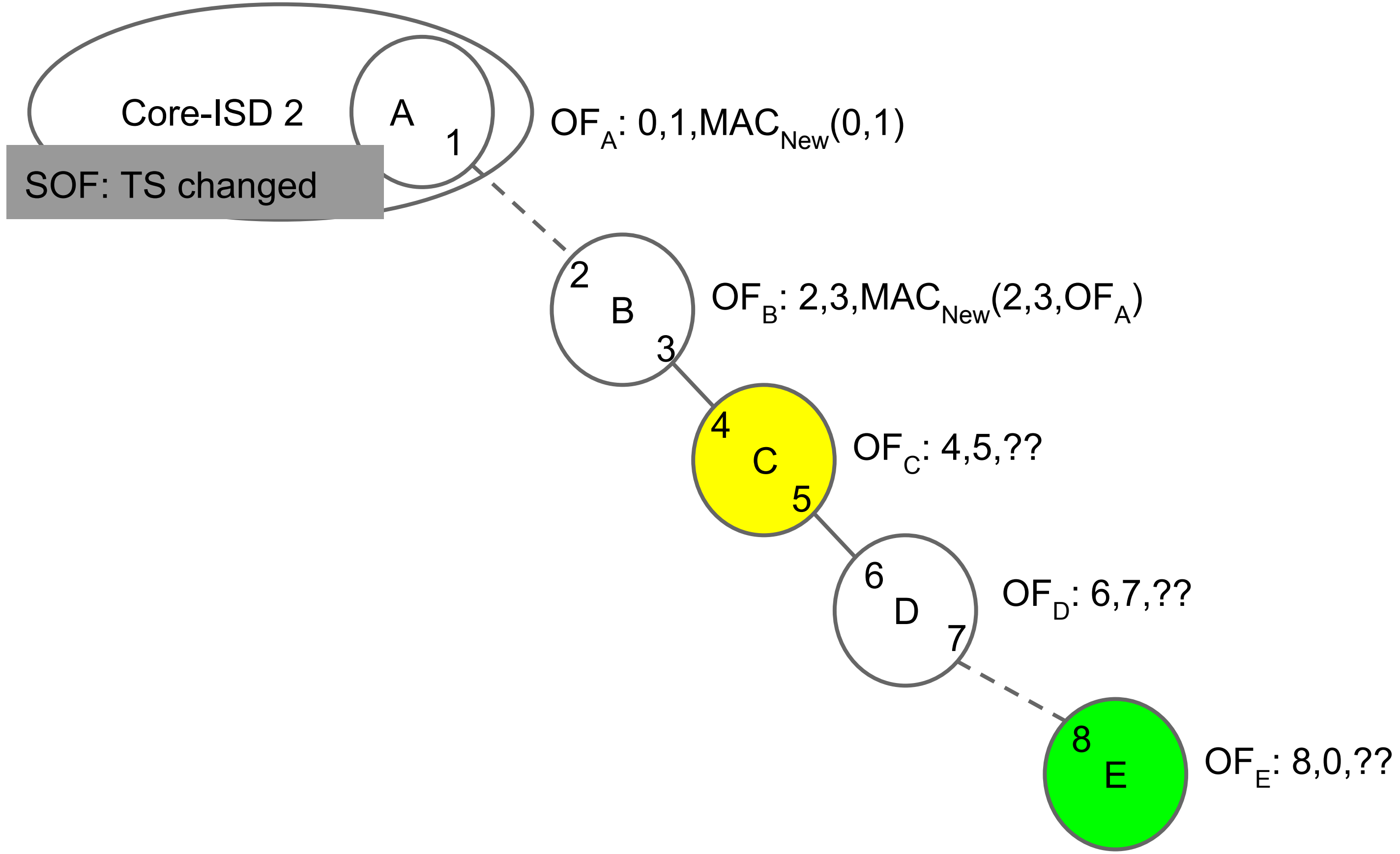
Attack Setting



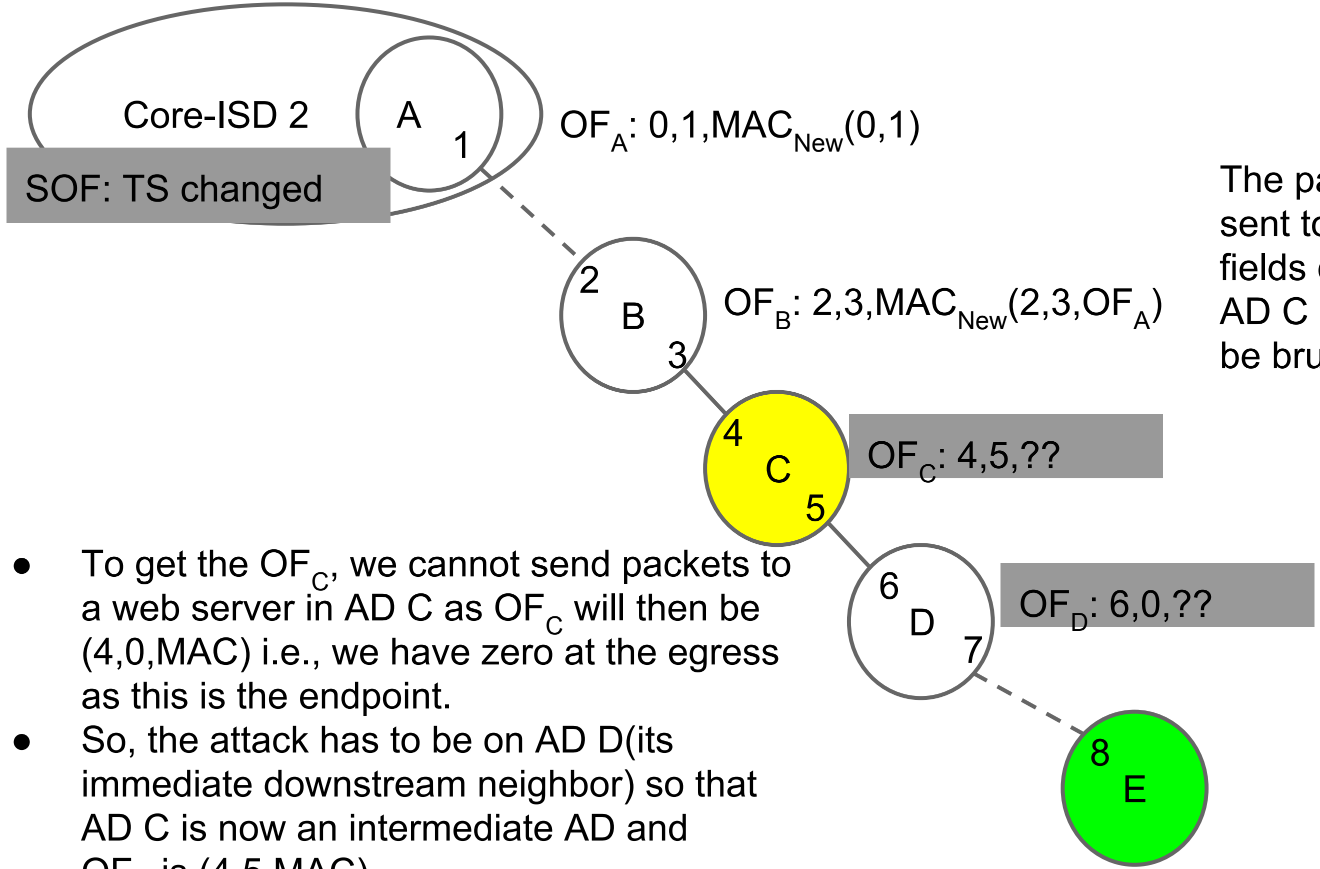
The attack

- Change of TS in down-path’s SOF means that the attacker will have to change all the opaque fields on down-path
- The attack would involve many sub-attacks targeting the AD’s in down-path in a top to bottom fashion, as each OF depends on previous OF
- In each sub-attack, the attacker will have to find a web server in the AD to be attacked and send brute-forced packets to it

Obtaining a new Opaque field



Obtaining C’s Opaque field



The packets are being sent to AD D. Opaque fields corresponding to AD C and AD D are to be bruteforced

- To get the OF_C, we cannot send packets to a web server in AD C as OF_C will then be (4,0,MAC) i.e., we have zero at the egress as this is the endpoint.
- So, the attack has to be on AD D(its immediate downstream neighbor) so that AD C is now an intermediate AD and OF_C is (4,5,MAC)

Attack complexity

- For a MAC size of 24 bits the attack size is $2^{24}*2^{24} = 2^{48}$
- The same attack can be repeated to obtain the rest of opaque fields in the down path
- It can be observed that once such an attack is performed, the same opaque fields can be used for duplicating some other down paths

Possible defenses

- Increase the complexity of the attack by using the one byte at the start and end of an up path/down path(e.g., egress field at end of down path) to extend the MAC size
- This will increase the attack complexity to 2^{56} as obtaining some OF’s involves brute forcing more bits
  - Specifically, getting the second opaque field and the opaque field of the AD just above the end point will have 2^{56} complexity

Attack using shortcut paths

- Till now, we have shown the attacker to be in a different ISD but this is not necessary
- Similar attack is possible by an attacker in the victim’s ISD using shortcut paths
  - Even in case of a shortcut path, we still have Time stamp field in SOF at the start of down path which is changed by the attacker

Thank You!!

Special thanks to Tae-Ho Lee, Samuel Hitz, Adrian Perrig