PIT matching from unregistered remote faces: a critical NDN vulnerability

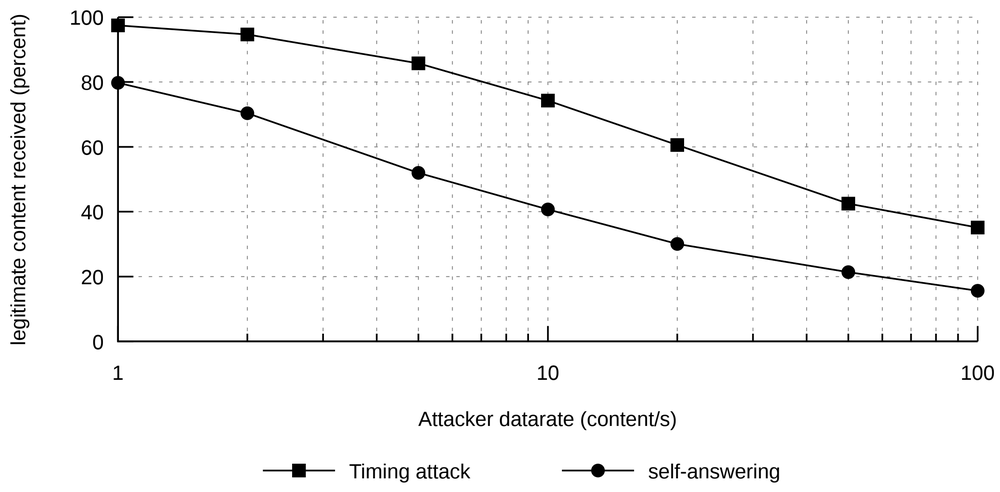
Xavier Marchal, Thibault Cholez, Olivier Festor
LORIA, UMR 7503 (University of Lorraine, CNRS, INRIA)
Vandoeuvre-les-Nancy, F-54506, France
{xavier.marchal, thibault.cholez, olivier.festor}@loria.fr

Introduction to the vulnerability

- Reference implementation lacks verification on incoming *Data* packets:
  - Only performs **scope check** and **PIT matching**
  - No verification of ***Data* packets incoming face**
- Malicious users may **not follow the standard**:
  - Send *Data* packets without receiving *Interest* packets
- This vulnerability enables attacks like:
  - **Denial of Service** by consuming PIT entries with fake *Data* packets
  - **Cache Poisoning**

```
void
Forwarder::onIncomingData(Face& inFace, const Data& data) {
  {...} //some stuff and scope check
  //PIT match check
  pit::DataMatchResult pitMatches = m_pit.findAllDataMatches(data);
  if(pitMatches.begin()==pitMatches.end()){
    //goto Data unsolicited pipeline
    this->onDataUnsolicited(inFace,data);
    return;
  }
  // CS insert
  m_cs.insert(data);
  {...} //and so on...
}
```

Effect of the attacks on the client



- Experiments were done with:
  - C's data-rate of 4 *Interest*/s with a Zipf distribution for 100 contents
  - 100ms latency between R1 and S
  - Legitimate *Data* packet freshness of 0ms and fake *Data* packet freshness of 0ms for timing attack and 2s for self-answering
- A can highly reduce the number of legitimate *Data* packets C receives depending on the attack aggressiveness

Remediation using Interest outgoing Face(s)

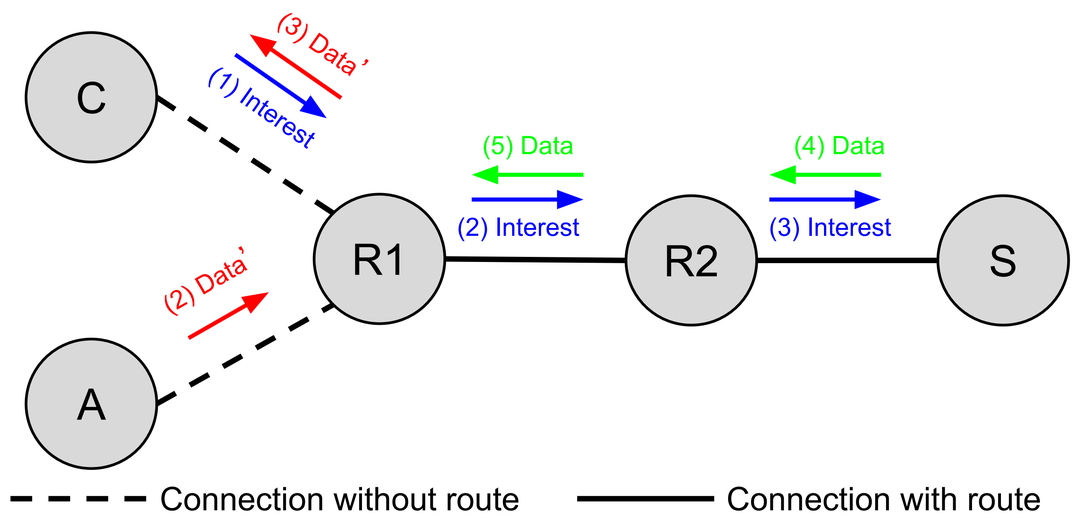
- Drop *Data* packet if it does not come from a face to which a corresponding *Interest* packet was forwarded
- Outgoing faces are **already present** in NFD but only for NAcks
- Advantage: Simply **extends** the usage of outgoing faces for *Data* packet

```
{...} //after PIT match check
auto it = pitMatches.begin();
while(it != pitMatches.end()){
  if((*it)->getOutRecord(inFace) == (*it)->out_end()){
    it = pitMatches.erase(it);
  } else{
    ++it;
  }
}
{...} //but prior Unsolicited Data pipeline
```

Attack topology

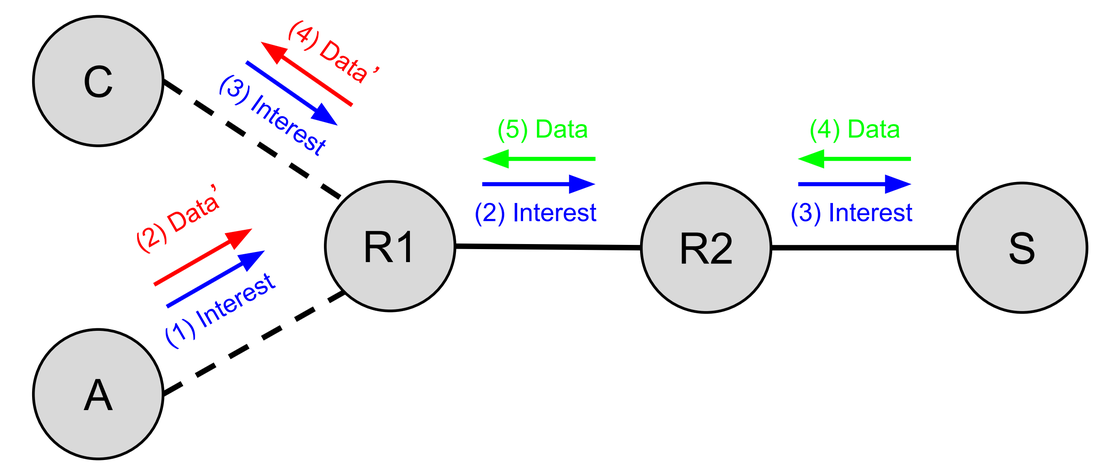
- **C** is a Client that wants to access a resource
- **A** is a malicious user that wants to disturb the network
- **R1** is an access router, for example the router right after a DSLAM
- **R2** is a core router
- **S** is the legitimate producer of the resource

Timing Attack



- A sends fake *Data* packets to consume legit PIT entries
- A can't see other clients' traffic but can increase its success rate by:
  - focusing on popular contents
  - increasing its data-rate

Self-answering attack



- A sends *Interest* packet immediately followed by *Data* packet that can match it
- A can easily insert any *Data* packets in the Content Store

Remediation using FIB entries

- Check if the *Data* packet came from a face that can forward *Interests* packet with the same prefix; Else drop the *Data* packet
- Better than *Interest* outgoing faces for **multicast** strategies
- Advantages: **stateless** and applicable prior to PIT lookup

```
{...} //some stuff and scope check
auto& fibEntry = m_fib.findLongestPrefixMatch(data.getName());
if (!fibEntry.hasNextHop(inFace)){
  return;
}
{...} //PIT match check, CS insert and so on...
```