Design of a Stream-based IP Flow Record Query Language

Vladislav Marinov, Jürgen Schönwälder

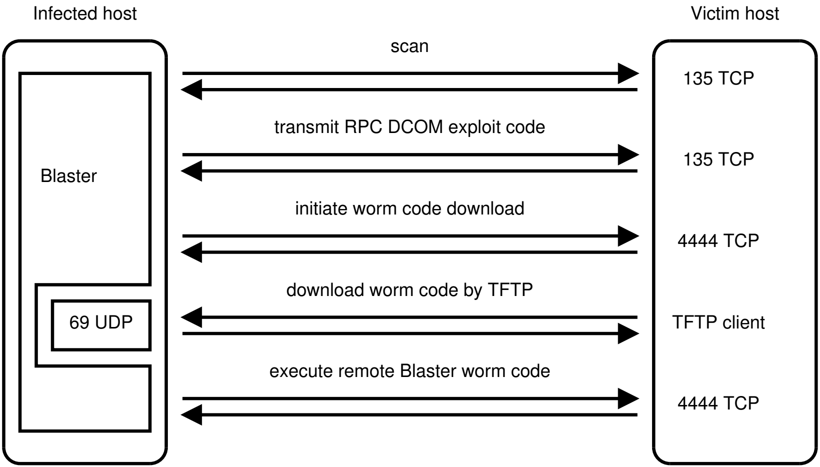
JACOBS UNIVERSITY

DSOM 2009, Venice, 2009-10-27

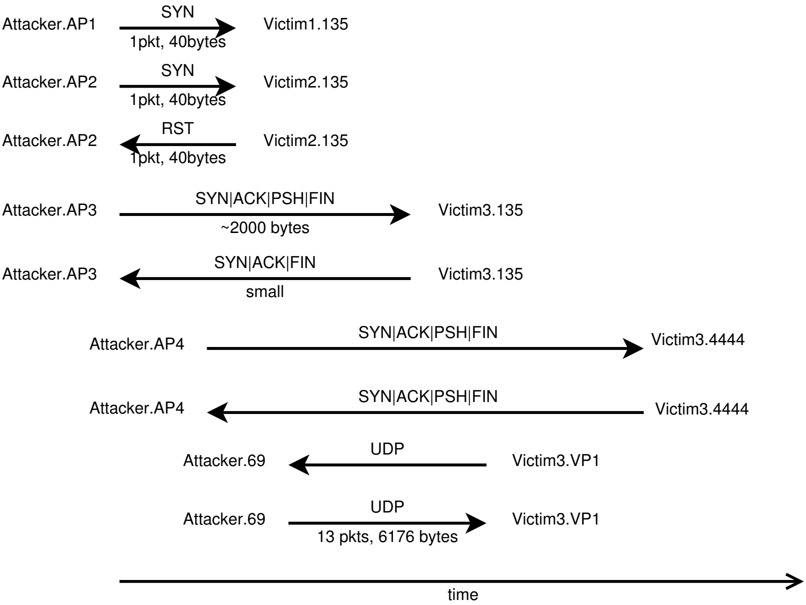
Support: EU IST-EMANICS Network of Excellence (#26854)

Outline of the Talk

Blaster Worm Infection



Blaster Worm Infection Flow Pattern



State of the Art and Problem Statement

Existing Query Languages

Existing Approaches

- SQL-based languages such as Gigascope [1] and Tribeca [2] can lead to poor query performance or with SQL optimizations to poor insert performance
- Filtering languages such as those used by nfdump, flow-tools [3], CoralReef [4] lack a time and concurrency dimension
- Procedural query languages such as those used by NeTraMet, flow-tools, Stager [5], Silk [6] are powerful but not trivial to understand

Conclusion

- Existing languages cannot describe traffic patterns composed of a set of flows that have time dependencies

Problem Statement and Approach

Problem Statement

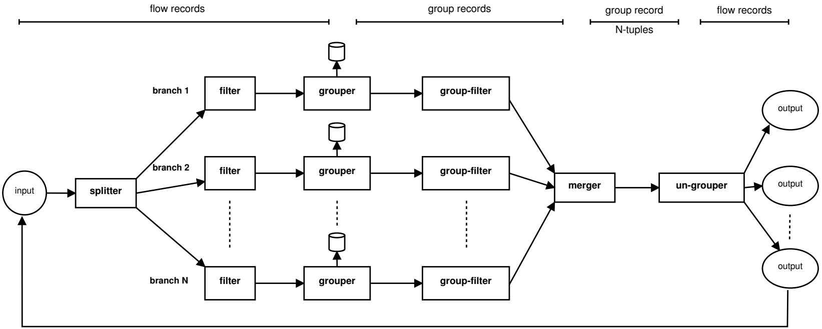
- Describe and identify the occurrence of network traffic patterns in a collection of flow records
- Describe the timing (causal) relationships between flows involved in a pattern

Approach

We propose a new IP flow record query language to describe network traffic patterns in a declarative and easy to understand way using Allen's time interval algebra.

- A comprehensible set of language primitives will be built
- The flow patterns of some common network services and application will be derived

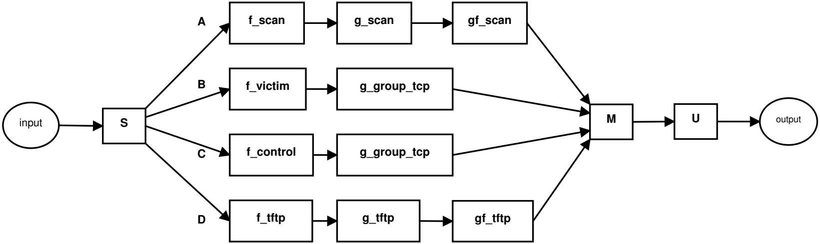
Framework for IP Flow Filtering



- Stream-based approach with several operators
- Primitives to express timing and concurrency relationships
- Primitives to define dependencies among flow attributes

Blaster Worm Example Continued

Blaster Worm Infection Query



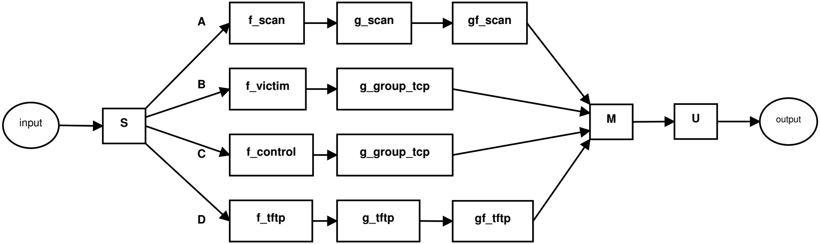
A branch capturing the scanning (TCP/135) acitivity

B branch capturing the exploit (TCP/135) activity

C branch capturing the control (TCP/4444) activity

D branch capturing the TFTP (UDP/69) download

Wiring of the Query Operators




```
splitter S {}

ungrouper U {}

input -> S
S branch A -> f-scan -> g-scan ->gf-scan -> M
S branch B -> f-victim -> g-group-tcp -> M
S branch C -> f-control -> g-group-tcp -> M
S branch D -> f-tftp -> g-tftp -> M
M -> U -> output
```

A: Capturing Scanning Activity

```
filter f-scan {
    dstport = 135
    proto = tcp
    flags = S
}

group-filter gf-scan {
    count > 20
}
```

```
grouper g-scan {
  module g1 {
      srcip = srcip
      dstip = dstip relative-delta 1
      stime = stime relative-delta 5ms
      stime = stime absolute-delta 5s
  }
  aggregate srcip, union(dstip),
            min(stime), max(etime),
            count
}
```

- The filter f-scan selects TCP/135 flows
- The grouper g-scan groups all filtered flows with consecutive destination IP addresses within a maximum absolute time of 5s an a relative time delta of 5ms
- the group filter requires that a group of scanning flows contains at least 20 flows

B & C: Capturing Exploit and Control

```
filter f-victim {
    srcport = 135 OR dstport = 135
    proto = tcp
}

filter f-control {
    srcport = 4444 OR dstport = 4444
    proto = tcp
}
```

- The filter f-victim selects TCP/135 flows
- The filter f-control selects TCP/4444 flows
- The selected flows can have arbitrary TCP flags

B & C: Grouping TCP Flow Records

```
grouper g-group-tcp {
  module g1 {
   srcip = dstip
   dstip = srcip
   srcport = dstport
   dstport = srcport
   stime = stime relative-delta 5ms
  }
  module g2 {
   srcip = srcip
   dstip = dstip
   srcport = srcport
   dstport = dstport
   stime = stime relative-delta 5ms
  }
  aggregate g1.srcip as srcip,
            g1.dstip as dstip
            min(stime) as stime,
            max(etime) as etime
}
```

- Group “forward” and “reverse” TCP flows (g1)
- Group multiple flow records for the same flow (g2)
- Aggregate carries srcip and dstip (from g1) plus a meaningful start and end timestamp

D: Capturing TFTP Download

```
filter f-tftp {
    srcport = 69
      OR dstport = 69
    proto = udp
}

group-filter gf-tftp {
    bytes > 6K
}
```

```
grouper g-tftp {
  module g1 {
    srcip = dstip
    dstip = srcip
    srcport = dstport
    dstport = srcport
    stime = stime relative-delta 5ms
  }
  module g2 {
    srcip = srcip
    dstip = dstip
    srcport = srcport
    dstport = dstport
    stime = stime relative-delta 5ms
  }
  aggregate g1.srcip as srcip,
            g1.dstip as dstip
            min(stime) as stime,
            max(etime) as etime,
            g2.sum(bytes) as bytes
}
```

M: Merging Branches

```
merger M {
    A.srcip = B.srcip
    A.srcip = C.srcip
    A.srcip = D.dstip
    B.dstip = C.dstip
    B.dstip = D.srcip
    B.dstip in union(A.dstip)
    A < B OR A m B OR A o B
    B o C
    D d C
}
```

- A $<$ B: A (scan) before B (exploit)
- A m B: A (scan) meets B (exploit)
- A o B: A (scan) overlaps B (exploit)
- B o C: B (exploit) overlaps with C (control)
- D d C: D (tftp transfer) during C (control)

Conclusions

Conclusions

Contribution

- Design of a new stream-based flow query language supporting operators to express timing relationships between flows
- Demonstration of language features using the Blaster Worm Infection

Current Status and Future Work

- First implementation (flowy) written in Python completed
- Flowy performance analysis and optimization
- More advanced query analysis and optimizations
- Distributed processing of queries using mediators

References

C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk.
Gigascope: A Stream Database for Network Applications.
In *Proceedings of SIGMOD'03*, pages 647–651, New York, NY, USA, 2003. ACM.

M. Sullivan and A. Heybey.
Tribeca: a System for Managing Large Databases of Network Traffic.
In *Proceedings of ATEC'98*, pages 13–24, Berkeley, CA, USA, 1998. USENIX Association.

S. Romig.
The OSU Flow-tools Package and CISCO NetFlow Logs.
In *Proceedings of LISA '00*, pages 291–304, Berkeley, CA, USA, 2000. USENIX Association.

D. Moore, K. Keys, R. Koga, E. Lagache, and KC. Claffy.
The Coral Reef Software Suite as a Tool for System and Network Administration.
In *Proceedings of LISA'01*, pages 133–144, Berkeley, CA, 2001. USENIX Association.

A. Oslebo.
Stager-A Web Based Application for Presenting Network Statistics.
In *Proceedings of NOMS'06*, 2006.

Michael Collins, Andrew Kompanek, and Timothy Shimeall.
Analysts' Handbook: Using SiLK for Network Traffic Analysis.
CERT, 0.10.3 edition, November 2006.

T. Dübendorfer, A. Wagner, T. Hossmann, and B. Plattner.
Flow-level Traffic Analysis of the Blaster and Sobig Worm Outbreaks in an Internet Backbone.
In *Proceedings of DIMVA'05*, Vienna, Austria, July 2005. Springer's Lecture Notes in Computer Science (LNCS 3548).