Using Argus and Postgres to analyse network flows for security

David Ford
OxCERT
Oxford University Computer Services

What are network flows?

- A convenient way of representing traffic on your network
- Contain a timestamp, the source/destination IP, protocol/port, traffic volumes, and a status (eg RST, CON, TIM)
- One flow may represent many packets
- Do **not** contain the packet payloads

Why would I want to use them?

- Good for understanding security incidents after they've happened - how did an attacker get in? what else did the attacker compromise?
- Can help you to identify suspicious/abusive behaviour
- Can help in tracing other network issues (eg tracing the source of load on a particular link)

What is argus?

- Which Argus? we mean: (http://www.qosient.com/argus) - Audit Record Generation and Utilisation System
- It can capture from a live interface (eg a mirror/span port, or a fibre tap), or from a Cisco netflow source, or from a pcap file, and indirectly from other sources such as sFlow

What is argus (2)

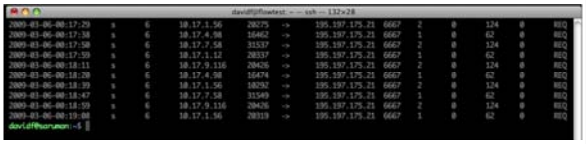
- It stores data in its own record format, and contains tools to extract data as required
- Most of the tools for using it are command line driven, but can easily be automated to produce useful reports, or to extract the data you need
- Syntax for extracting data is very similar to that used by tcpdump/wireshark

Where to capture?

- Depends largely on your network topology, what you want to see, and how much data you wish to collect
- Things to consider include:
  - locations of NATs - do you want to see traffic before, or after NATing
  - firewalling - do you want to see traffic that gets through the firewall, or traffic that doesn't
  - router locations - will you miss internal data if you only capture at the border

A warning about NATs

- even having flows from before and after NATing does not guarantee you can trace the source of a malicious flow
- If a single destination host has both malicious and non-malicious connections from behind the NAT, it may not be possible to distinguish these without logs of NAT translations
- Problem cases include IRCds, Virtual Hosted websites etc.



```
2009-03-06-00:17:29    s    6    10.17.1.56    20275  ->    195.197.175.21  6667   2    0    124   0    REQ
2009-03-06-00:17:38    s    6    10.17.4.98    16462  ->    195.197.175.21  6667   1    0    62    0    REQ
2009-03-06-00:17:50    s    6    10.17.7.58    31537  ->    195.197.175.21  6667   2    0    124   0    REQ
2009-03-06-00:17:59    s    6    10.17.1.12    20337  ->    195.197.175.21  6667   1    0    62    0    REQ
2009-03-06-00:18:11    s    6    10.17.9.116   20426  ->    195.197.175.21  6667   2    0    124   0    REQ
2009-03-06-00:18:20    s    6    10.17.4.98    16474  ->    195.197.175.21  6667   1    0    62    0    REQ
2009-03-06-00:18:39    s    6    10.17.1.56    10292  ->    195.197.175.21  6667   2    0    124   0    REQ
2009-03-06-00:18:47    s    6    10.17.7.58    31549  ->    195.197.175.21  6667   1    0    62    0    REQ
2009-03-06-00:18:59    s    6    10.17.9.116   20426  ->    195.197.175.21  6667   2    0    124   0    REQ
2009-03-06-00:19:08    s    6    10.17.1.56    20319  ->    195.197.175.21  6667   1    0    62    0    REQ
davidf@saruman:~$
```

How much data

- Argus records can be compressed (using gzip) - our experience suggests this can cut the size requirements by a factor of 5-10 (or more in some cases - particularly for regular scanners)
- Data size will depend a lot on how many sources and how many flows you are recording - for our systems, this equates to roughly 15GB/day compressed, however you may be able to reduce this substantially

Understanding your data sources

- It’s important to understand how various different types of packet flows are handled by your flow capturing devices
- Dependant on network configuration, the equipment/protocol you are using to capture flows. For example a scan of unresponsive hosts may be recorded as INT, or TIM/RST this behaviour may make it easier or harder to distinguish successful connections

data sources (part 2)

- How do your capture devices cope if they receive too many flows to process, do they sample the data - if so, how?, do they stop passing packets in the case of a router, or do they stop recording flows
- Remember the point where you receive unexpectedly large numbers of flows is probably the point you want to have all the flows to work out why!

What can you do with the flows?

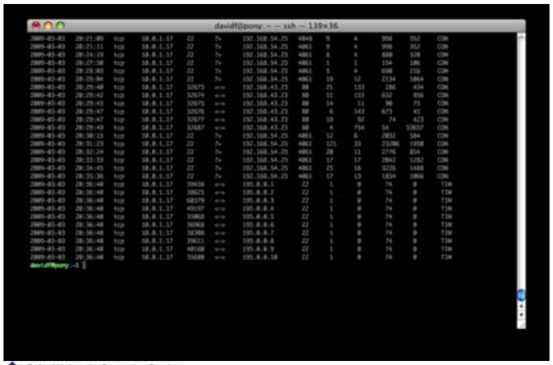
- Incident investigation - how did a machine get hacked, from where?
- Spotting malicious hosts, P2P, other rogue traffic
- Identifying hosts talking to known bad guys

Incident Investigation

- Starting point here is that we know that a particular machine has been compromised (possibly through other flow analysis, or possibly because we've been alerted to it from elsewhere)
- We want to know:
- how was it compromised, and when
- did the attackers get in anywhere else?
- which remote hosts are taking part, so that we can identify other hosts affected

A simple example

- We receive a notification (at 8am) that a host (10.0.1.17) is scanning out on port 22.
- We don't know at this stage when the system was compromised, and we can't tell whether the logs will give us any clue
- But network flows show (appropriately anonymised):



davidf@pony: ~ — ssh — 139×36

```
2009-03-03  20:21:09  tcp   10.0.1.17  22     ?>   192.168.54.25  4048  9    4    998    352    CON
2009-03-03  20:21:11  tcp   10.0.1.17  22     ?>   192.168.54.25  4061  9    4    998    352    CON
2009-03-03  20:24:19  tcp   10.0.1.17  22     ?>   192.168.54.25  4061  8    4    888    328    CON
2009-03-03  20:27:38  tcp   10.0.1.17  22     ?>   192.168.54.25  4061  1    1    134    106    CON
2009-03-03  20:28:03  tcp   10.0.1.17  22     ?>   192.168.54.25  4061  5    4    698    216    CON
2009-03-03  20:29:04  tcp   10.0.1.17  22     ?>   192.168.54.25  4061  19   12   2134   1064   CON
2009-03-03  20:29:40  tcp   10.0.1.17  32673  <->  192.168.43.23    80  21   133   286    434   CON
2009-03-03  20:29:42  tcp   10.0.1.17  32674  <->  192.168.43.23    80  11   133  632     956   CON
2009-03-03  20:29:45  tcp   10.0.1.17  32675  <->  192.168.43.23    80  14    11   98      75   CON
2009-03-03  20:29:47  tcp   10.0.1.17  32676  <->  192.168.43.23    80  6    143  673      41   CON
2009-03-03  20:29:47  tcp   10.0.1.17  32677  <->  192.168.43.23    80  19    52    74    423   CON
2009-03-03  20:29:49  tcp   10.0.1.17  32687  <->  192.168.43.23    80  4    754  54    53657   CON
2009-03-03  20:30:15  tcp   10.0.1.17  22     ?>   192.168.54.25  4061  12   6    2032   584    CON
2009-03-03  20:31:23  tcp   10.0.1.17  22     ?>   192.168.54.25  4061  121  33   23306  1950   CON
2009-03-03  20:32:24  tcp   10.0.1.17  22     ?>   192.168.54.25  4061  28   11   2776   854    CON
2009-03-03  20:33:33  tcp   10.0.1.17  22     ?>   192.168.54.25  4061  17   17   2042   1282   CON
2009-03-03  20:34:45  tcp   10.0.1.17  22     ?>   192.168.54.25  4061  25   16   3226   1488   CON
2009-03-03  20:35:36  tcp   10.0.1.17  22     ?>   192.168.54.25  4061  17   13   1854   1066   CON
2009-03-03  20:36:40  tcp   10.0.1.17  39438  <->  195.0.0.1        22  1    0    74     0       TIM
2009-03-03  20:36:40  tcp   10.0.1.17  38625  <->  195.0.0.2        22  1    0    74     0       TIM
2009-03-03  20:36:40  tcp   10.0.1.17  60379  <->  195.0.0.3        22  1    0    74     0       TIM
2009-03-03  20:36:40  tcp   10.0.1.17  49297  <->  195.0.0.4        22  1    0    74     0       TIM
2009-03-03  20:36:40  tcp   10.0.1.17  35068  <->  195.0.0.5        22  1    0    74     0       TIM
2009-03-03  20:36:40  tcp   10.0.1.17  36968  <->  195.0.0.6        22  1    0    74     0       TIM
2009-03-03  20:36:40  tcp   10.0.1.17  38308  <->  195.0.0.7        22  1    0    74     0       TIM
2009-03-03  20:36:40  tcp   10.0.1.17  39611  <->  195.0.0.8        22  1    0    74     0       TIM
2009-03-03  20:36:40  tcp   10.0.1.17  40568  <->  195.0.0.9        22  1    0    74     0       TIM
2009-03-03  20:36:40  tcp   10.0.1.17  35608  <->  195.0.0.10       22  1    0    74     0       TIM
davidf@pony:~$
```

- We can see that the system first began scanning out at 20:36:40, and that shortly before this there were several connections in on port 22 from 192.168.54.25
- looking back we can see connections from this IP, and another one earlier in the evening
- We can also see some port 80 traffic to 192.168.43.23 which looks as though it may have been initiated after the connections
- We now have timestamps and malicious host names - use this to hunt through argus logs and host logs

Spotting malicious traffic as it happpens

- We've so far been looking at data that's been collected and archived
- We can also analyse live data to identify unexpected traffic patterns such as scanners, P2P users, botnets etc.
- Older Argus versions (2.0.5) came with an example perl script to do this - you may wish to write your own as it hasn't been updated for a while

- The aim is to import argus data as it is recorded and to look for patterns such as repeated connections out to different hosts on a single port (scanning), or huge numbers of inbound connections
- The "holy grail" could be some way to track connection behaviour for hosts against past traffic patterns, however I'm not aware of any such scripts for argus
- You could also check the flows against a list of known bad hosts - or this could be done overnight

Argus Issues

- Dataformat issues in argus 2.0.6 - designed for 32-bit platforms only - don’t switch platform and expect data to be reliably readable, and don’t use argus 2.0.6 on AMD64
- For a new deployment you almost certainly want to use argus 3 - the datafiles are much saner
- IPv6?

Argus Issues (2)

- On a large site like ours, with lots of data passing through the main router, processing the logs is slow
- If we discover a new host we need to investigate it could take 20-30 minutes to get all the flows extracted we need
- some data is commonly accessed for many types of incident

Use of a database

- We find it helpful to put some of our argus data into a relational database
- Tables can get quite large so being selective as to what is most useful is important
- We are using a system with Postgres as an SQL backend, with some perl/PHP scripts to import/extract data

- As an example you might wish to put TCP traffic into your database - you might also want to remove some other very common ports to keep table sizes managable
- Our model is to use one table per day
- Even with these restrictions expect multi Gigabyte tables - We find 4-6GB to be typical
- We import data every 30 minutes (when we rotate argus data files), and index only when the day’s table is filled (indexed on ip and port)
- if you wish to do daily reporting, you can do it once the indexes are built

- Once indexed tables exist you can extract other data and do other analysis eg. packet counts, volumes per IP
- you may wish to store this data in your database for longer than the full flow data
- graphing, trend analysis etc.

- In fact, the Argus developers have been working on SQL support. Initial code was released in early March 2009
- Their code deals with some potential issues with ICMP flows (which we’ve never dealt with in our DB)
- Their code currently targets MySQL however in either case the benefits of using a database should be similar

Other potentially useful Argus related tools

- Arguseye - a GUI for certain Argus tasks. Could be useful for extracting data and investigating an incident (requires Argus 3)
- Flowscan - can produce graphs from various types of network flow sources. Reportedly supports Argus

Conclusions

- Network flow analysis is useful for both security and other purposes
- Argus can help capture, collate and process flow data
- For large volumes of data you may find storing your data in a database improves performance

Questions?

- Thanks to:
- Robin Stevens (OUCS, OxCERT)
- Jonathan Ashton (OUCS, OxCERT)
- Oliver Gorwits (OUCS)
- Patrick Green (Warwick)