Providing process origin information to aid in network traceback

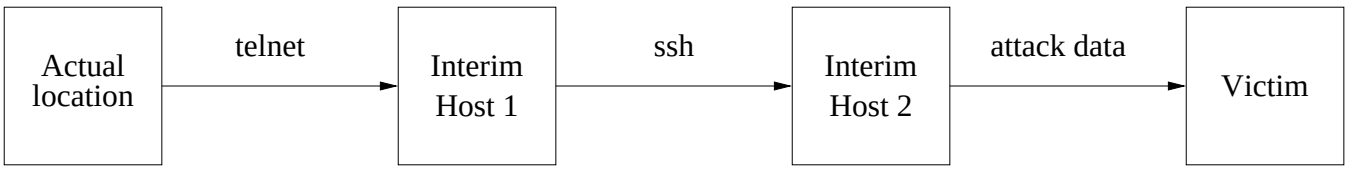
Florian Buchholz
PI: Prof. Clay Shields
Purdue University, CERIAS

Network traceback: Packet marking

- IP datagrams can be easily spoofed to hide the real address from where it originated.
- Attackers do not want to reveal their own location or the location of hosts they have compromised.
- Idea: supply destination of the datagram with data from the routers that were used to forward the datagram.
- Either *mark* the datagram itself or send separate control message to the destination.

Network traceback: Stream correlation

- Tries to correlate streams of TCP connections observed at different points in the network architecture.
- An interactive connection can extend over a chain of multiple hosts.



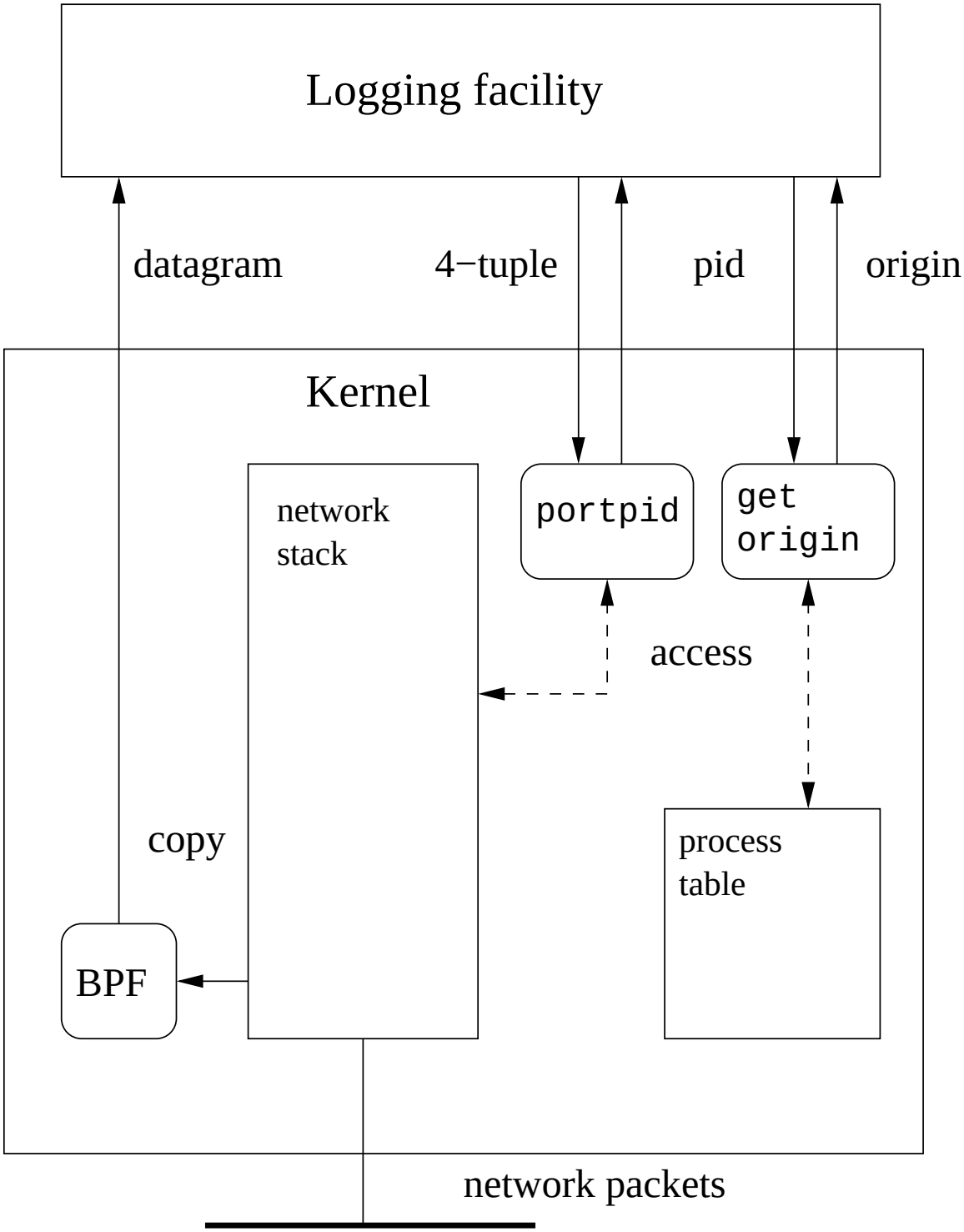
- Focus lies on what can be be deduced from information obtained from various fixed checkpoints in the network

Host causality

- Traffic entering a host on the network can be transformed in such a matter that it can no longer be related to traffic leaving it.
- Datagrams can be delayed in a host so that no immediate connection between incoming and outgoing packets can be made.
- A new area of research that is concerned with data transformations or data flow tracking on a host is needed.
- No research in this area has been pursued in connection with the traceback problem thus far.

Process origin

- Determine whether a process was started locally or from a remote shell. Store location as origin information.
- If a process that sends out network traffic and is of remote origin, one can then make a connection between the datagrams that were sent out and the actual origin of the process.
- Incoming datagrams for a process can also be associated, both with the origin itself and possibly outgoing datagrams.
- It does not matter, whether there is a delay or transformation of the data that enters the system before it leaves the host.

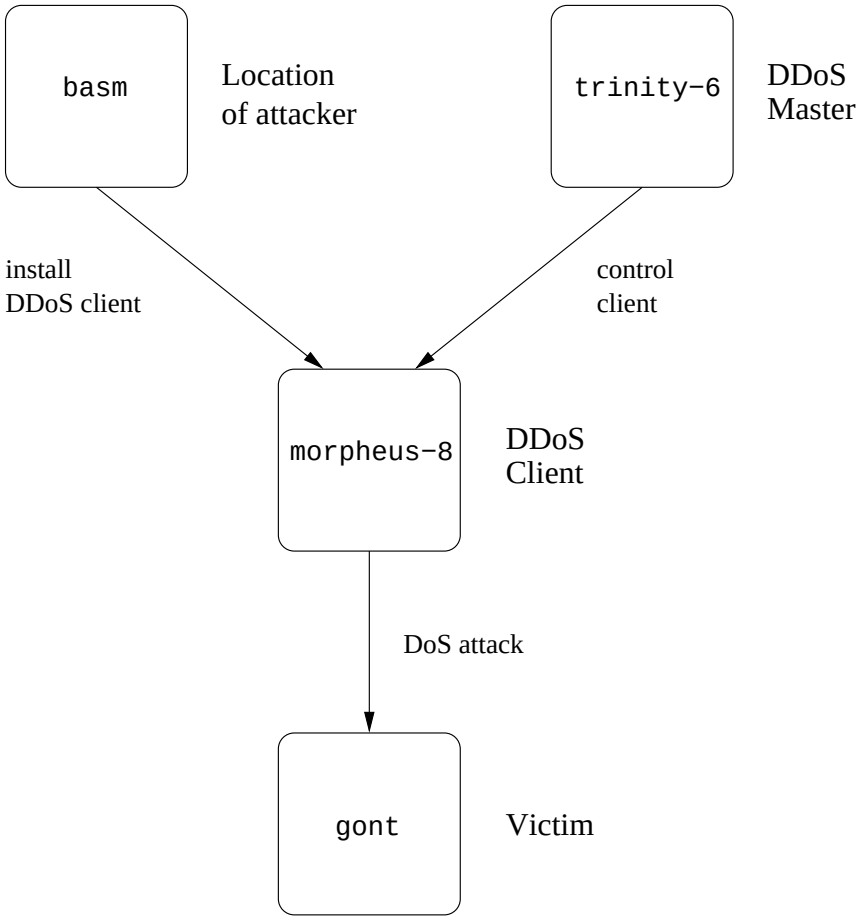

Logging facility
datagram
4−tuple
pid
origin
Kernel
network
stack
portpid
get
origin
access
copy
process
table
BPF
network packets

Example 1 (Stepping Stone)

In this example, `morpheus-8` was used as a stepping stone. A user from `basm` (`128.10.243.21`) logged into `morpheus-8` (`128.10.251.107`) via `ssh`. From there, he used `ssh` again, to log into `lisa.cs` (`128.10.7.22`). The logging facility recorded the following entry from this:

```
morpheus-8:1022->lisa.cs:22 sent by pid 285
Origin: basm:1022-morpheus-8:22
```

Example 2 (Trinoo client)



Example 2 (cont.)

```
morpheus-8:1117->trinity-6:31335 (17) sent by pid 3760
Origin: basm:32155-morpheus-8:13419

trinity-6:39805->morpheus-8:27444 (17) received by pid 3760
Origin: basm:32155-morpheus-8:13419

morpheus-8:1135->gont:12865 (17) sent by pid 3760
Origin: basm:32155-morpheus-8:13419
morpheus-8:1135->gont:59850 (17) sent by pid 3760
Origin: basm:32155-morpheus-8:13419
morpheus-8:1135->gont:10435 (17) sent by pid 3760
Origin: basm:32155-morpheus-8:13419
morpheus-8:1135->gont:4577 (17) sent by pid 3760
Origin: basm:32155-morpheus-8:13419
```

Effects on system behavior

- Most lookups involve only simple pointer lookups and the copying of the data structures.
- The call to `portpid` is in its essence what the IP packet demultiplexing code in the networking stack does. For TCP SYN packets and UDP packets, there is two times the demultiplexing effort.
- In every call to `accept`, the `lastaccept` field is set from the socket information. This is a cheap copy operation.
- Whenever a process spawns a child process, the `origin` field is copied. This is part of a copy operation that is done anyway.

Conclusions

- The system works well under the given model
- There is little overhead compared to normal system behavior
- Information is protected in kernel
- Process origin can lead to better mechanisms in file systems and access control
- Problems: `cron`, batch jobs and startup scripts