Obeseus – a lightweight DDOS detector for big attacks

1 We all have failings

When I built the Internet Barometer, I realised there was a flaw. The system had many memorable qualities:

- It was built on Snort so it could take advantage of a huge amount of signatures
- Unlike standard Snort, our snort system boasted some nice bespoke enhancements including asynchronous IO, offline session reconstruction and a more advanced pcap module.
- With these customizations we bundled the usual go-faster stripes, like MMAP, PF_RING and buffer tuning.

And the end result of this was a Snort instances that was pretty much capable of 1Gbs. We then implemented this software on special hardware which comprise an FPGA IO-processor coupled to the NIC allowing us to ***pre-qualify*** packets before they burn resources on the BUS & CPU, multiple CPU's allowing the load to be spread over many Snort instances and RAMDISKs to reduce IO wait time took us to a fully capable 10Gbs implementation. Link the lot together with some cross-compilers and a control system, and you have something pretty impressive, certainly better than any commercial offering.

But there is at least one flaw, with Snort as a detection engine - it can't detect DDOS attacks.

2 The Theory according to LFB – in brief

Snort is good at looking atomic attacks – those attacks that take advantage of a bug, weakness or miss-configuration of a networked device sent in a single packet. To use the correct jargon, a ***Threat Agent*** (aka hairy bad guy) is attempting to ***exploit*** a known vulnerability. Snort is even quite good at detecting these vulnerability attacks that span over a number of packets, although it must be said some tools do the necessary normalising, defragmentation and session reassembly better.

These attacks are often design to gain unauthorised access to computer resource. Everybody will remember the host of Windows RPC exploits that have endured the last decade. Conversely, some of these attacks can be DOS attacks – ***Denial of Service*** attacks or sometimes known ***as Disruption of Service*** Attacks. These, like the above, are launched from one computer manned by the same hairy bad guy against a specific target computer, with the intent of making that target computer unusable – typically by making it hang or loop. Common and rather historical attacks are:

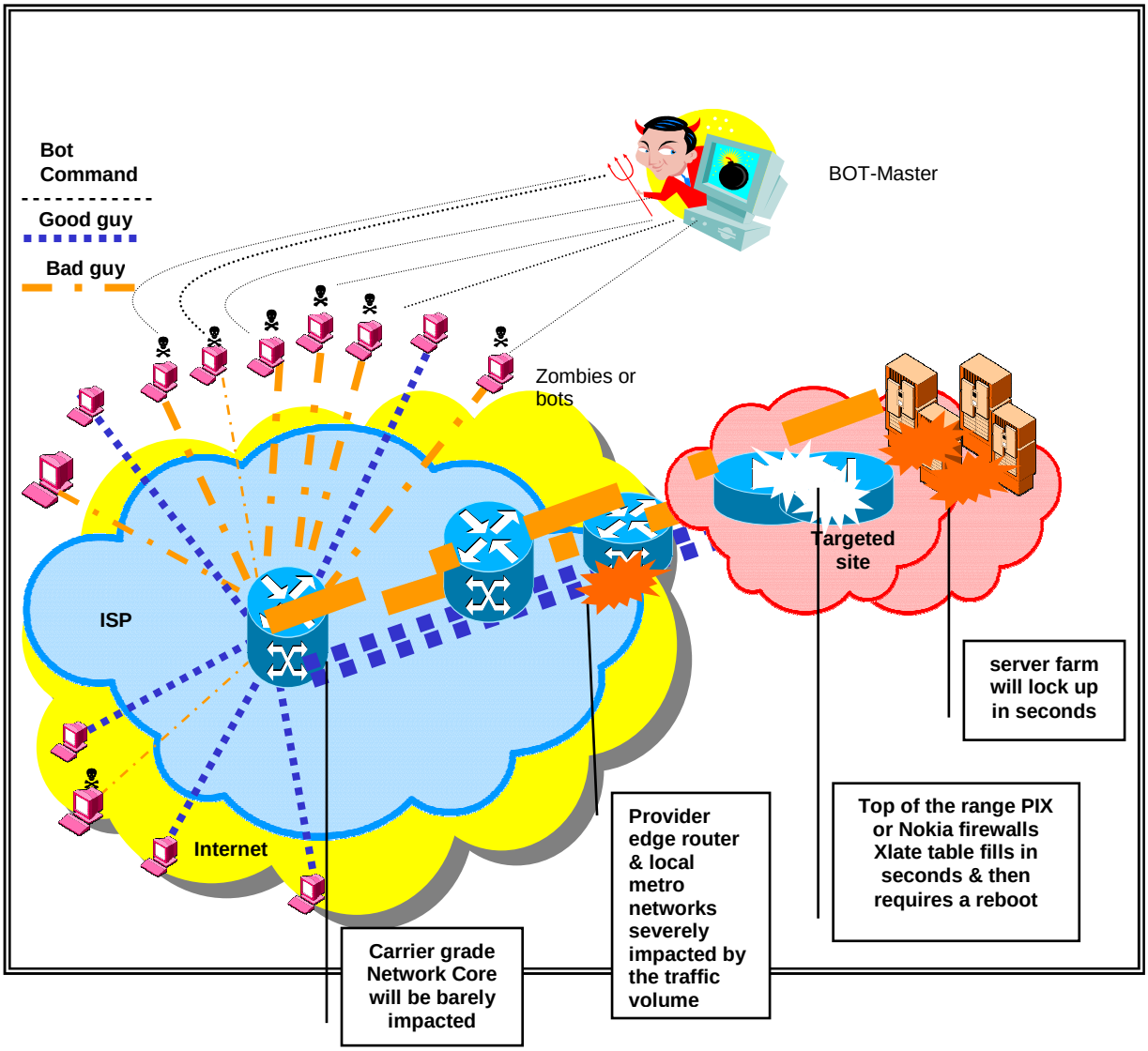
- Syn-flood
- Smurf

- Papa-smurf
- Nuke
- Fraggle
- Evil ping

Many of these attacks can't work with one packet – they are more effective at burning-up the resources of the target computer if they are repetitive in their nature. They overwhelm the target by sending a flood of attacks. However, an attack launched by one malevolent domestic computer that relies on power can easily be defeated by the power and advanced techniques available to a modern enterprise. These attacks needed to evolve to survive – so they did.

And so did the purpose of their use – rather than the preserve of hairy hackers, the "Flood Attack 2.0" is the homeland of organised crime replete with its own payment systems and rules. More commonly known as *a* ***Distributed Denial of Service*** attacks, this must be the first practical misuse of Grid Computing. The bad guys have truly embraced the idea of flooding and to a degree abandoned the idea of exploiting a vulnerability and use the power of 10,000's, 100,000's and even millions of home computers to overwhelm the target with sheer power –Grid Computing!!!. These source machines (aka BOT or Zombie) usually with out-of-date virus protection systems tend to work fine at their day-to-day tasks despite the fact that they have been infected. That is until the evil BOT-MASTER sends them a command code. At that time, these bots devote all their power to sending attack packets at the target contained in the command code.

Small attacks present no problem to a large network provider – their impact will be restricted to the targets' equipment and may be handled by a customer who has purchased adequate bandwidth, properly configured firewalls or servers. Large attacks are very different. The result is a domino effect – the targeted servers & firewall collapse rapidly as shown below.



The upstream edge router and local metro network switch which are sized to handle a smaller amount of geographic traffic will continue to deliver a percentage of the traffic but will struggle with the load. Even if an out-of-band management network is in place, the CPU will be sky-high and the management services on the router will be unusable making it impossible to determine the cause – and without adequate information, technicians will be generating hypothesis about routing failure, bugs or any other causes. And of grave concern to the ISP, this edge equipment which is used to distribute traffic to many customers in that geo-location and probably will service 10's or 100's of local customers – not just the targeted customer – will be providing sub-optimal performance. Revenues can be severely impacted as SLAs are breached.

3 So what do I want from my new DDOS detection SW

Obviously, I have a number of commercial DDOS detectors and having used them, I believe in many situations, there would be a place for something different.

Most of these commercial tools keep a big database with traffic profiles to identify the attacks. With lots of work and careful tuning, they can be used to determine even the smallest attack. All very commendable But Why ?

As explained before, Little attacks on modern infrastructure will have little impact. We really don't want to spend too much time on these – the effect is small, limited to one customer and so will be easily identified with the normal telemetry which remains functioning. We want to concentrate on big attacks - big attacks have big impacts. And surely these big attacks are noticeable without detailed profiling. Surely base determina can be used?

The other problem these commercial tools is their fixation with flows as a means of rationalising the measurement of traffic. I think this derived from the use of netflow and s-flow, which are a means of accounting. It superimposes the concept of a bi-direction of flow onto two independent transmissions between two peers. These Netflow/S-flow records are ideal for the billing accounting purposes for which they were designed and as they are available "free of charge" is a great convenient source of data but there are drawbacks. These are:

- They are sampled over relatively large time periods;
- The records have few fields which restricts attack analysis
- Converting data to flows loses information & increases reaction time
- The processing is not real-time

This is a brief 2 page explanation of the general subject – if you enjoyed it, you may enjoy the more lengthy coverage in *How to Cheat at Managing Information Security* published by Syngress. For a more stern, trite and glossy coverage, most security vendors will have some details on the subject.

4 So what will ours do

Our objective was clear, we wanted to build a system that would detect notable flood attacks – Those big attacks that would severely impact more than one customer.

We already had an extensive multi-probe infrastructure coupled to a central analysis server which we were very happy with as a platform. Apart from this, every other design constraint/objective was open. After some consideration, this is what we came up with.

Each probe:

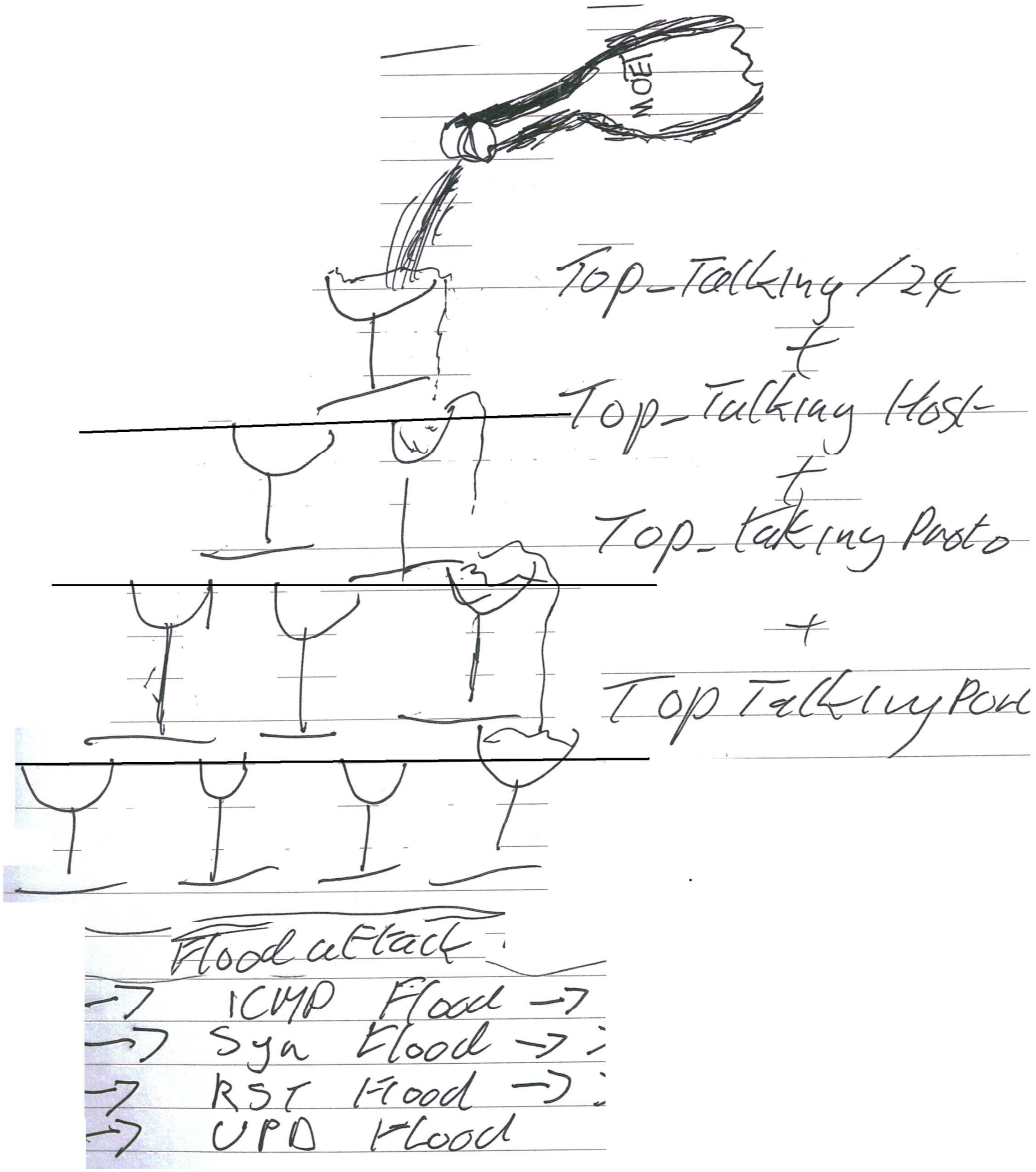
- Would not alert based on historical profiles;
- Would use protocol analysis and rate anomalies to determine an attack, using the whole packet;
- Would accumulate a number of these factors in a “balanced score-card” scoring system for each of the multiple criteria applied;
- Would be state-less – not attempting to reconstruct any flows ;
- Would use basic math & algorithms and be light-weight so that it can be ported to an FPGA resident version;
- Would be self-learning with black lists.

The central analysis server would also use a “Balanced Score-card” scoring system and combine a number of techniques including

- Multi-point analysis – verification from multiple probes
- Possible use of cusum analysis or rolling exponential
- Use of commonly published Internet backlists

5 How will ours do it

I was thinking how to crack the problem in the worlds most boring meeting and realised I had doodled the solution.



The processing would be as follows:

1. Do initial processing to establish the speed and volume of traffic passing across the interface. This will be used in the next stage.
2. Group all traffic travelling past the interface into evenly sized buckets and accumulate the traffic into each bucket appropriate to its destination address. The bucket size defaults to a CIDR /24 but the bucket size and sample size are determined in point 1. For the top N buckets…..
3. Determine the top talked-to Host in the bucket.
4. Determine the top talked-to protocol on the Host
5. Determine the top talked-to port on the Host

At each stage, we generate a BPF command and a snort command to generate the traffic capture required (for the eventual FPGA resident version).

Once we have determined the exact destination address, protocol and port, the system analyses the traffic one more time. During the whole process, the system captures the following information.

- The ratio of the top protocol to the rest
- The ratio of SYN packets to other packets
- The ratio of SYN packets to FIN packets
- The ratio of RST packets to other packets
- The ratio of RFC1918 packets to other packets

Further more, it will record the top (n) talking source addresses – this isn't really used at the moment but in the future it could be used to block traffic or do more advanced source address analysis.

Because the probe has access to the whole packet, it calculates the number of packets that have identical payload, a sure sign of automated flooding.

6 So what does it detect

We have it running parallel with a commercial product and it detects the following attacks

- SYN floods
- RST floods
- ICMP floods
- General UDP floods
- General TCP floods

7 Where does it go from here

As stated, the current software is implement entirely on x86 Linux. The intention is to port much of the code to the FPGA power NIC on our probes. We will make this original Linux software generally available after the FPGA port.

Another feature for the roadmap is to expand the capability from being a detector to being a scrubber/cleanser. We have a suitable software router that can be integrate with the existing code to re-router cleaned traffic.

Lastly, we would like to see if can integrate existing protocol analysis engines from other products as plug-ins so we can validate traffic, without re-inventing the wheel.