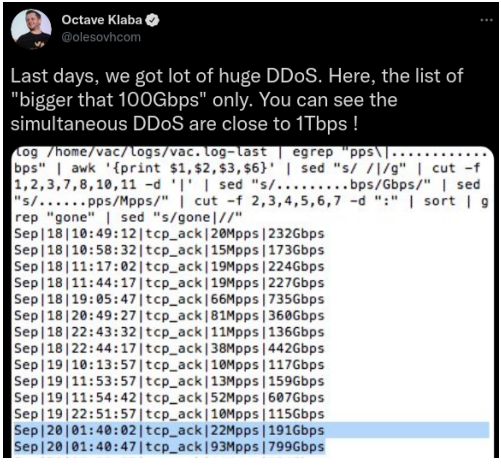
LUND UNIVERSITY

X-Pro: Distributed XDP Proxies against Botnets of Things

SYAFIQ AL ATIIQ, CHRISTIAN GEHRMANN
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY, LUND UNIVERSITY

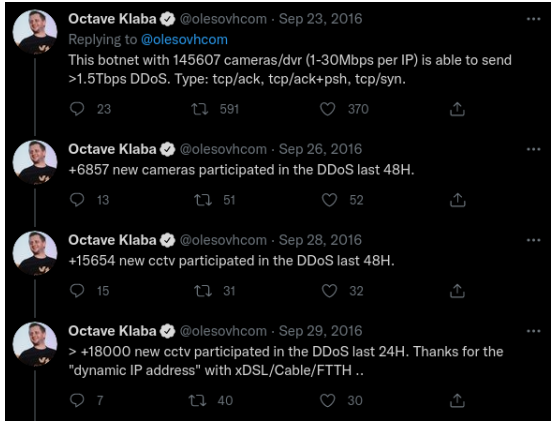
Table of Contents

What happened in 2016? [1]



[1] https://twitter.com/olesovhcom/status/778830571677978624

What are the sources of the attack? [2]



[2] https://twitter.com/olesovhcom/status/778830571677978624

And the follow ups...[3] [4]

KrebsOnSecurity Hit With Record DDoS

September 21, 2016 122 Comments

On Tuesday evening, KrebsOnSecurity.com was the target of an extremely large and unusual distributed denial-of-service (DDoS) attack designed to knock the site offline. The attack did not succeed thanks to the hard work of the engineers at **Akamai**, the company that protects my site from such digital sieges. But according to Akamai, it was nearly double the size of the largest attack they'd seen previously, and was among the biggest assaults the Internet has ever witnessed.

[3] https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/

[4] https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/

Later, it is known as: Mirai

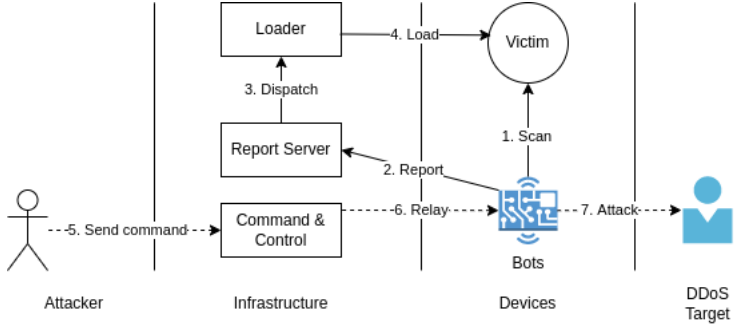


Figure: Mirai Operation[5]

[5]Manos Antonakakis, et.al. 2017. Understanding the mirai botnet. In Proceedings of the 26th USENIX Conference on Security Symposium (SEC'17). USENIX Association, USA, 1093–1110.

Attack Strategies

Two types of DDoS strategy:

1. Periodically-low-rate attack: behaves identically to the regular traffic pattern. The intention is to fly under the radar, while consuming the victim's resources.
2. Massive attack: large amount of traffic at once.

Both types are considered to be harmful, either for the target nodes or for the IoT devices, as they are usually battery-driven and resource-constrained. X-Pro proposes a solution to this problem.

Why bother providing "yet another" DoS countermeasure?

Traditionally, DoS is handled at the victim-end or core-end, applying network-level detection and filtering. We have observed that this traditional way of handling DDoS does not consider the new IoT communication patterns, i.e. :

- IoT devices typically do not primarily communicate with general internet services but are directed towards a specific backend.
- Botnet threats on IoT entities are undesirable from a resource perspective, and this gives a large incentive for IoT device owners to implement DDoS countermeasures at the source not just on the network level.

What is XDP? [6]

- Stands for eXpress Data Path, a novel programmable packet processing hook living inside the linux kernel-space.
- In XDP, the underlying operating system accommodates a safe execution environment to run an eBPF program.
- XDP provides a safe, fast, and customizable packet processing integrated with the kernel networking stack.

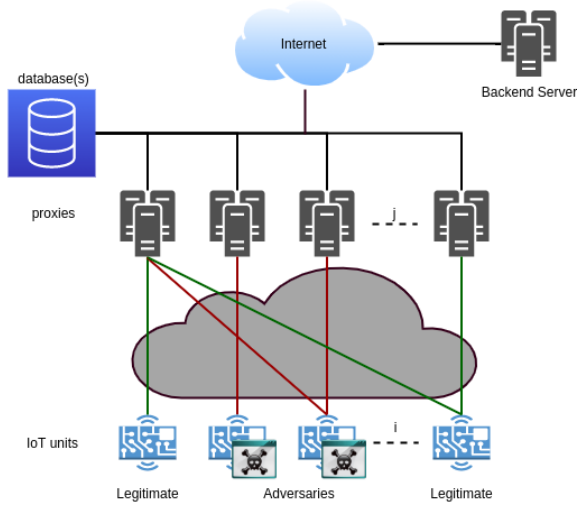
[6] Toke Høiland-Jørgensen, et.al., 2018. The eXpress data path: fast programmable packet processing in the operating system kernel. In Proceedings of CoNEXT '18.

Table of Contents

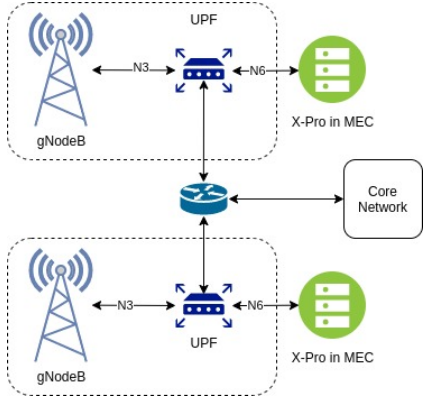
X-Pro

Under the assumption that **adversary can infect IoT units** but not prevent packets from flowing over the proxies, X-Pro prevents servers from being overwhelmed by DDoS by blocking the unwanted traffic with distributed proxies. To handle distributed DoS, information about traffic condition is shared between the proxies via a centralized database node.

X-Pro Overall Architecture



Where does X-Pro fit in the network?



X-Pro Filtering Design

- We use a filtering approach where each packet arrived at a particular proxy, is analyzed, and potentially blocked.
- Such mechanism is performed as early as possible, i.e., in the XDP hook.
- The blocking decision is based on a set of threshold parameters. These can be tuned to get the right trade-off between security and false blocking decisions.

Algorithm 1 Packet Filtering Procedures

1: <Lookup ts'_1, ts'_2, c for record (i, D_{addr}) in L_{DB} >
2: **if** record found **then**
3: **if** $t - ts'_2 > T_{T1}$ **then**
4: $ts'_1 = t, c' = 0, dc = 0, mark = 1$
5: **end if**
6: **else**
7: $ts'_1 = ts'_2 = t, c' = 0, dc = 0, mark = 0$
8: **end if**
9: $c' = c' + 1, dc = dc + 1, ts'_2 = t$
10: **if** $ts'_2 - ts'_1 > T_{T2}$ **then**
11: **if** $c'/(ts'_2 - ts'_1) > T_{F1}$ **then**
12: <Drop packet>
13: <Send an overload warning>
14: **end if**
15: **end if**
16: $ts_1* = \min_{u_i \in U_{D_{addr}}} {ts_1}'_i$
17: $ts_2* = \max_{u_i \in U_{D_{addr}}} {ts_2}'_i$
18: $c* = \Sigma_{u_i \in U_{D_{addr}}} (c_i + dc_i)$
19: **if** $ts_2* - ts_1* > T_{T3}$ **then**
20: **if** $c*/(ts_2* - ts_1*) > T_{F2}$ **then**
21: <Drop packet>
22: **end if**
23: **end if**
24: <forward packet>

U	Set of IoT units in the system
P	Set of proxies in the system
u_i	An IoT unit with index i
p_j	A proxy with address j
L_{DB}	A local database at p_j
D_{addr}	Destination IP address of a packet
ts'_1	Time stamp in p_j indicating the "oldest" packet time for a particular (i, D_{addr})
ts'_2	Time stamp in p_j indicating the "most recent" packet time for a particular (i, D_{addr})
c	Packet counter for a particular (i, D_{addr}) pair
dc	Delta packet counter (internally within a proxy) for a particular (i, D_{addr}) pair
T_{T1}	First filtering reset threshold used by a proxy
T_{T2}	Second Filtering minimum measure time threshold used by a proxy
T_{T3}	Third filtering minimum measure time threshold used by a proxy
T_{F1}	First packet maximum allowed frequency threshold used by a proxy
T_{F2}	Second packet maximum allowed frequency threshold used by a proxy

X-Pro Synchronization Design

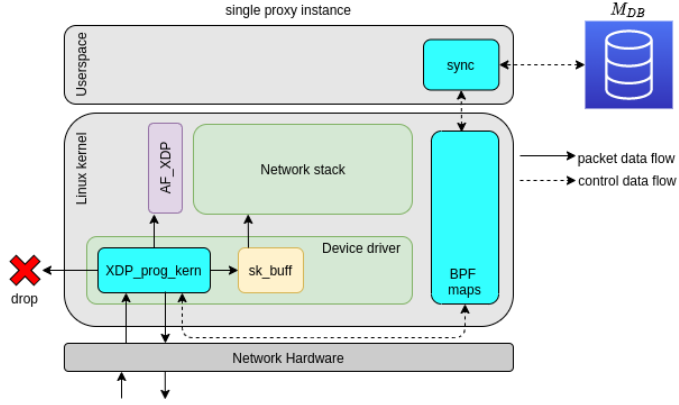
- To have the same visibility on each proxy, one needs to have a synchronization function between them.
- For every time period t, each proxy performs the synchronization procedure with the centralized database M_{DB}.
- The main challenge with the synchronization lies in the counts and count window in local db L_{DB} might be different from the centralized db M_{db} since the last synchronization. Hence, the synchronization must be able to cope with these changes.

Algorithm 2 Proxy Synchronization Protocol

1: p_j looks all the pair (i, D_{addr}), for $u_i \in U$
2: **for** each (i, D_{addr}) in M_{DB} **do**
3: **if** $L_{DB} \ni i, D_{addr}$ **then**
4: **if** $(mark' = 1)$ **then**
5: $< mark' = 0 >$
6: **if** $(ts'_1 - ts_1 > T_{T1})$ **then**
7: $< dc' = 0 >$
8: $< ts_1 = ts'_1, ts_2 = ts'_2, c = c' >$
9: **end if**
10: **else if** $(mark' \neq 1)$ **or** $(mark' = 1 \ \& \ ts'_1 - ts_1 < T_{T1})$ **then**
11: **if** $ts'_2 < ts_2$ **then**
12: $< ts'_2 = ts_2 >$
13: **else**
14: **if** $ts'_2 - ts_1 > T_{T4}$ **then**
15: $< ts'_1 = ts'_2 - (ts'_2 - ts_1)/r >$
16: $< c = \lfloor c/r \rfloor, ts_1 = ts'_1, ts_2 = ts'_2 >$
17: **end if**
18: $< ts_2 = ts'_2 >$
19: **end if**
20: **if** $ts'_1 > ts_1$ **then**
21: $< ts'_1 = ts_1 >$
22: **else**
23: **if** $ts'_2 - ts'_1 > T_{T4}$ **then**
24: $< ts'_1 = ts_1 >$
25: **else**
26: $< ts_1 = ts'_1 >$
27: **end if**
28: **end if**
29: $< c' = c + dc', c = c', dc' = 0 >$
30: **end if**
31: $< c = c', dc' = 0 >$
32: **else**
33: $< ts'_1 = ts_1, ts'_2 = ts_2 >$
34: $< c' = c, dc' = 0, mark' = 0 >$
35: **end if**
36: **end for**
37: **for** each (i, D_{addr}) in L_{DB} **do**
38: **if** $M_{DB} \ni i, D_{addr}$ **then**
39: $< ts_1 = ts'_1, ts_2 = ts'_2, c = c' >$
40: **end if**
41: **end for**

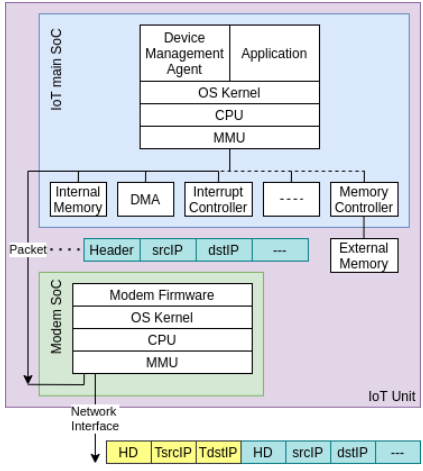
Table of Contents

Proxy



- Implemented on top of Linux Kernel 5.6 inside a Fedora 30.
- Running with 1 vCPU and 1024 MB mem.
- The packet filtering procedure is implemented as an eBPF program in `xdp_prog_kern`, and L_{DB} is stored in BPF maps. And there exist a daemon in the userspace to perform synchronization towards M_{DB}.

Device Side Design



Device Side Design

- The X-Pro design requires all traffic from the IoT units are routed through the proxy.
- An attacker aware of this principle can circumvent this mechanism by avoiding the whole proxy network, and fulfill the DoS target.
- Therefore, it is a mandatory design requirement for the IoT device to prevent its IP traffic control part from being infected by a malicious software.

Device Side Implementation

- ESP32 : 32 bit single-core (up to 240 MHz), 520 KiB SRAM, Wi-Fi 802.11 b/g/n.
- FiPy : Based on Espressif ESP32 SoC, 4MB RAM, 802.11b/g/n.

Device Side Implementation

- Prototype 1: ESP32 + FiPy (Connected to UART).
  ESP32 acts as the main SoC and FiPy acts as the modem SoC. Problem: UART is extremely slow!
- Prototype 2: ESP32 only.
  Main SoC and modem SoC logic are separated in a softwarized manner, i.e. : inside lwIP stack.

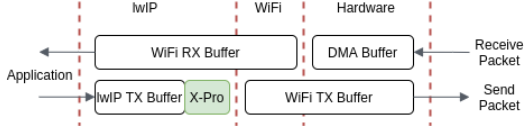


Table of Contents

Experimental Setup

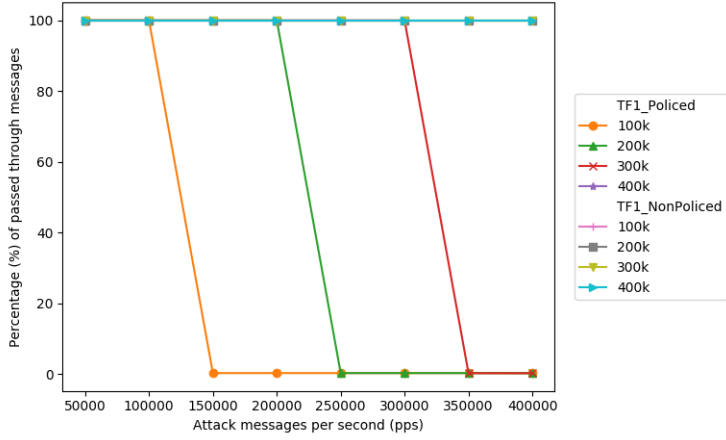
The proxies and the centralized database: Fedora 30, kernel version 5.6, one vCPU, 1024 MB memory.
How traffics are generated? PKTGEN[7]. We simulate the IoT units (infected and non-infected with botnets) with a Linux machine, running PKTGEN. PKTGEN sends CoAP messages, in which the size of each packet is 64 bytes. Message rates can easily be set through the ratep value in the pktgen configuration file.

[7] https://www.kernel.org/doc/Documentation/networking/pktgen.txt

Experimental Goals

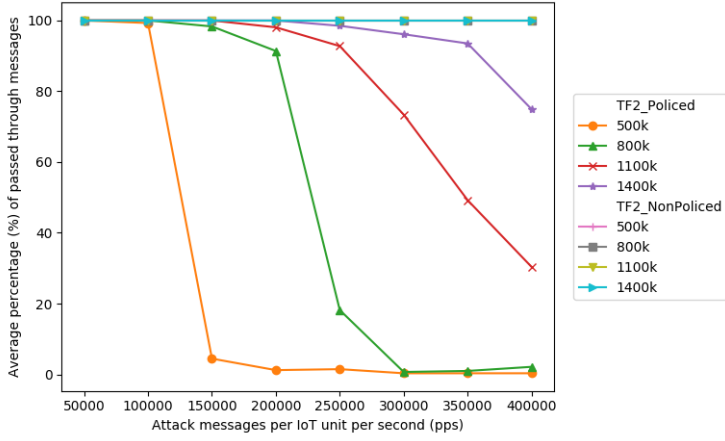
- In terms of the proxy: to measure how effective the suggested solution in terms of packet blocking for both single high-rate attack and low-rate attacks using fixed attack thresholds.
  - For high-rate attack, the setup consist of 1 proxy and 2 IoT units:
    - Non-malignant turned into botnet (policed flow).
    - Assumed to be always non-malignant (non-policed flow).
  - For low-rate attack, the setup consist of 4 proxies and 8 IoT units:
    - 4 non-malignant IoT units turned into botnets (policed flow), let's call it u_{even}.
    - 4 IoT units assumed to be always non-malignant (non-policed flow), let's call it u_{odd}.
    - A single proxy is connected to 1 u_{even} and 1 u_{odd}.
- In terms of the device side implementation: to measure the pure overhead at the device side for the two different implementation options.

Single Proxy



LUND UNIVERSITY

Multiple-Proxy Working Together



Device Side Measurement

Table: Overhead of the IoT Device

	Separated main MCU and network MCU	**lwIP logic of encapsulation**	
	Average RTT (ms)	**Average RTT (ms)**	**Average throughput (Mbits/s)**
X-Pro	50.55	37.67	25.86
vanilla	35.96	36.78	30.37

Table: Thingsboard with and without X-Pro

	Average RTT (ms)
Thingsboard + X-Pro	82.93
Thingsboard	80.98

Conclusions

- X-Pro is a distributed XDP proxies against botnets of things. The design of distributed proxies is armed with a centralized database, allowing the proxies to inform each other about the latest event in the networks. Through this collaboration, it is possible to defend the victim amid the situations when the adversaries are trying to send:
  - a massive and well-coordinated attack towards the victim through a single proxy.
  - periodically low-rate bogus messages spanned to multiple proxies in which the intention is to fly under the radar.
- The obtained results show that our solution allows strong protection of overload to both of the IoT backend and external attack targets.
- X-Pro requires the IoT units to be modified in regards to the network interface modem. As the adversary cannot tamper with the modification, they cannot re-route the destination of an outgoing packet, which always be forwarded to one of X-Pro's proxies.

Future Work

In future work, we will extend the solution with more advanced DDoS detection mechanisms (i.e., machine learning), which will allow automatic DDoS infection detection combined with efficient blocking.

LUND
UNIVERSITY