DEEPSEC

WEFF : P2P communication without 3rd party

Nikolaos Tsapakis, George Tselos

Who we are

- **Nikolaos Tsapakis** is a reverse engineering enthusiast and poetry lover from Greece. He is working as a security engineer. He has been writing papers or presented for Virus Bulletin, 2600, LeHack, Symantec, Hakin9, Athcon.

- **George Tselos** is a computer science tutor who lives and works in Athens, Greece. He is interested in embedded systems, microcontrollers and peripheral device development.

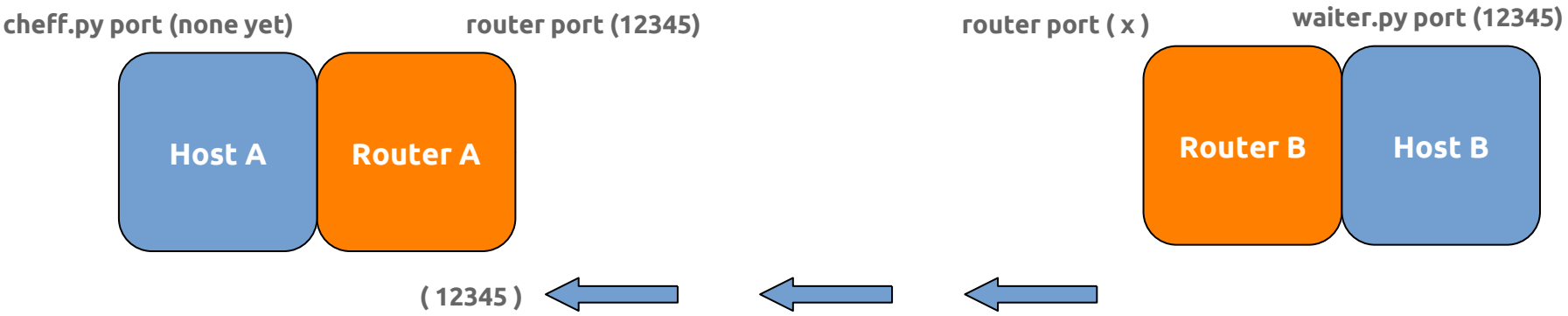
The problem

- p2p communication needs a 3rd party service to initiate like a stun server
- reason is routers and nats in network infrastructure block direct access to ports exposed to the internet (external ports)
- example of services that use 3rd party server are skype, zoom, viber
- why having a 3rd party subscription monitoring us ?
- https://www.cyberyodha.org/2023/04/what-is-stun-protocol.html

Solution

- brute force ports to trick application into establishing p2p comms
- both users are behind home routers
- WEFF - (w)aiter and ch(eff)
- B runs waiter.py, then A runs cheff.py
- p2p comms established, users chat through the program
- UDP protocol & AES encrypted communications
- python 3

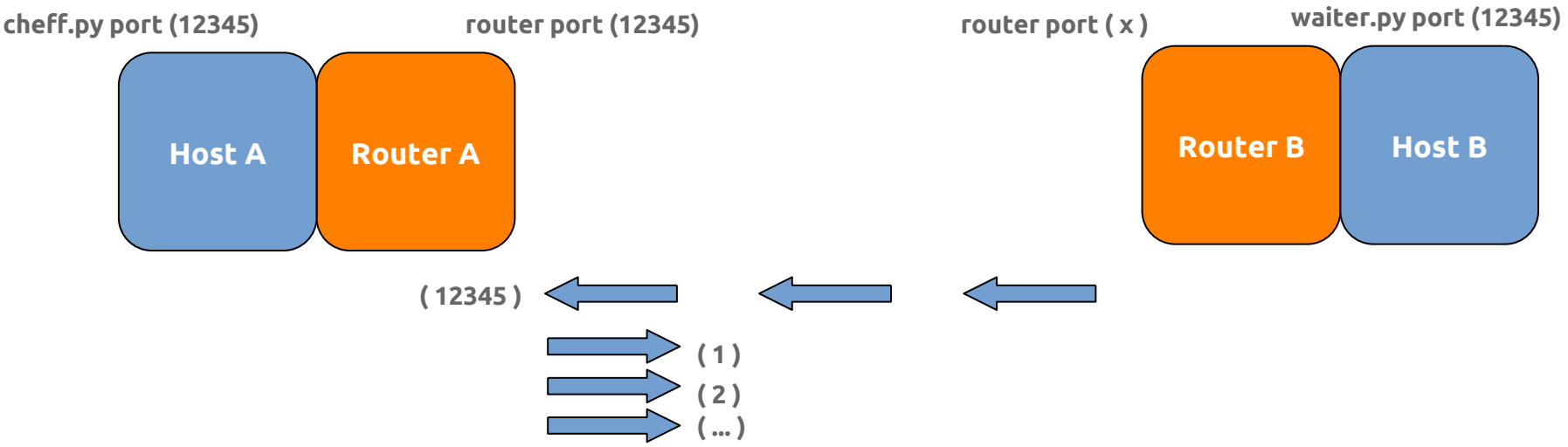
How it works



Waiter.py runs on B which periodically sends packets to A. This is to trick its own router B into allowing to receive incoming packets from router A, in following communication.

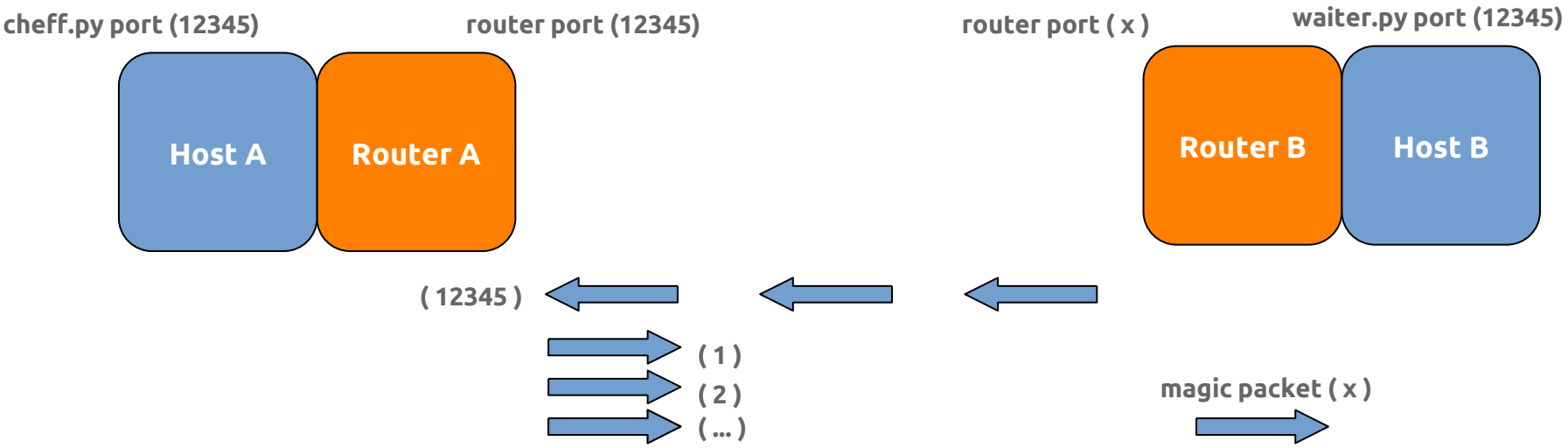
Waiter.py also sets application port on B to 12345 and destination port (router A port) to 12345. The router B port x is unknown at that time. Since cheff.py has not yet run on A there is no application port at that time on A.

How it works



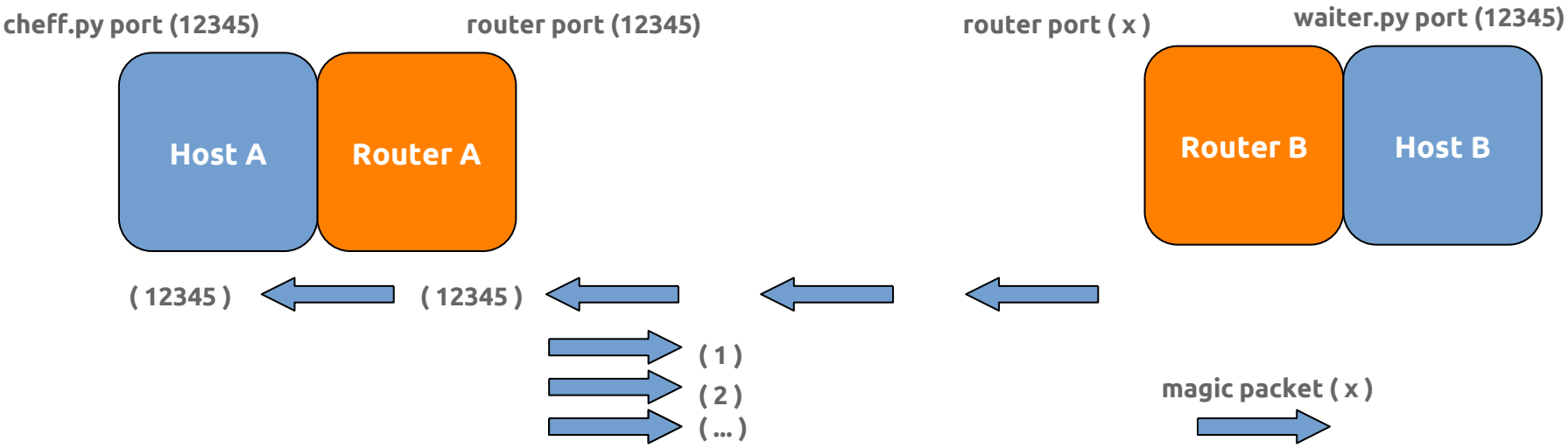
Cheff.py runs on A which starts sending packets to B sequentially from lowest to highest destination port number (1, 2, ..., x-1, x, x+1, ...), brute forcing the router B ports. Cheff.py on Host A sets application port to 12345.

How it works



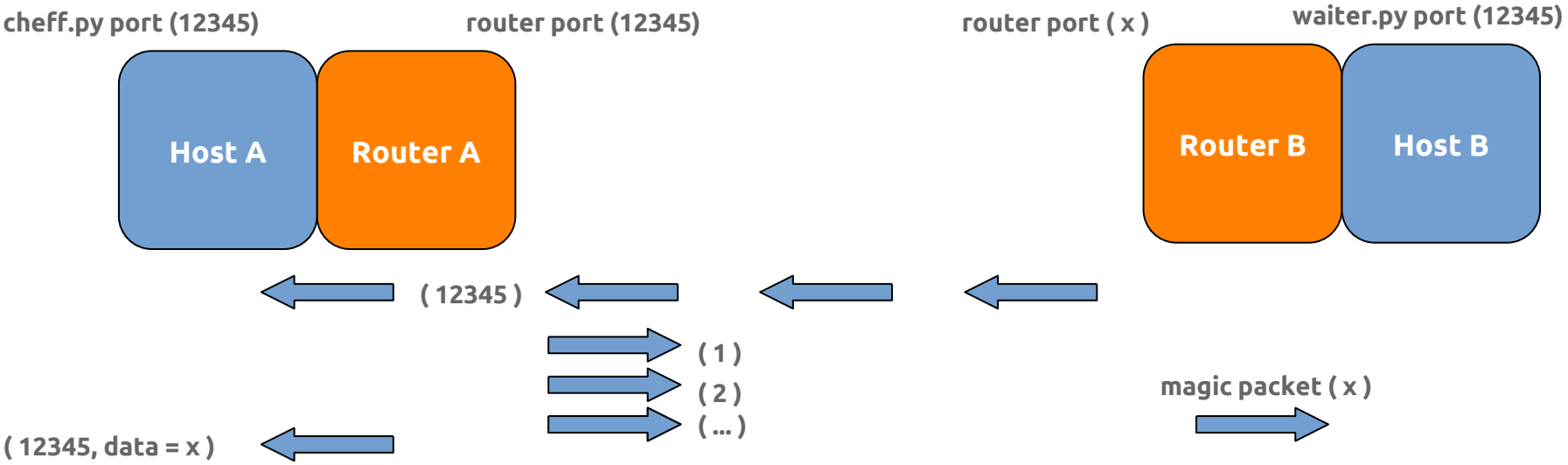
One of the brute force packets from A (magic packet) has same destination port number with listening router B port number (x). So, router B forwards that packet to waiter.py on host B.

How it works



At the same time, packet from B reaches cheff.py on A because the magic packet has just passed through router A and B, which means router A is able to process a packet from host B to host A as a reply.

How it works



Host B will extract destination port (x) from magic packet and will send it to host A with a new packet in its data. Host A now knows the destination port for host B (x) and will use it on cheff.py to send all future packets to B. Now cheff.py terminates the brute force process since port (x) is now known to A. Communications established.

Known limitations

- know IP for both hosts prior to establishing comms
- Tested only on home routers (1:1 port mapping on router A)
- delays while establishing communications
- network infrastructure related to UDP flood protection

Next steps

- only 1 ip known
- no need to brute force
- icmp over ipv6
- future work & presentation :)

Demo

- A, B on different geolocation and infrastructure
- A, B behind home routers
- presentation, demo, code at https://github.com/nitsa
- enjoy the demo

Q&A

Any questions ?

Thank you !