Fuzzing: Debrief

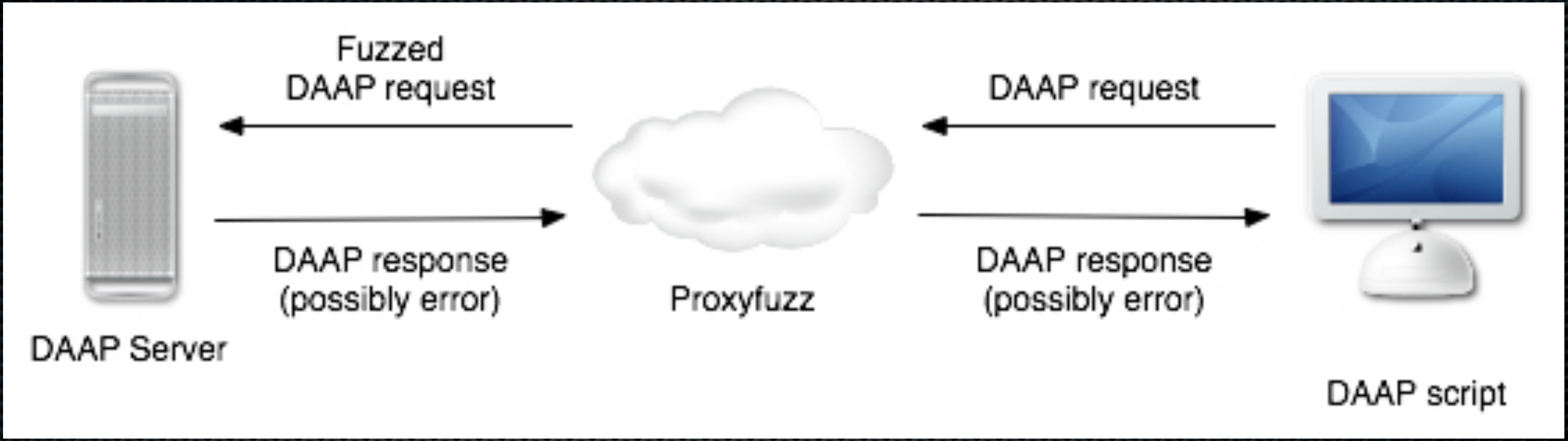
The Fuzzers

- ProxyFuzz
- General Purpose Fuzzer (GPF)
- Sulley

ProxyFuzz

- Python script which randomly inserts anomalies into network data
- Need a client which continuously generates data for the proxy to fuzz
- Completely unaware of protocol or condition of target

ProxyFuzz


Fuzzed DAAP request
DAAP request
DAAP response (possibly error)
Proxyfuzz
DAAP response (possibly error)
DAAP Server
DAAP script

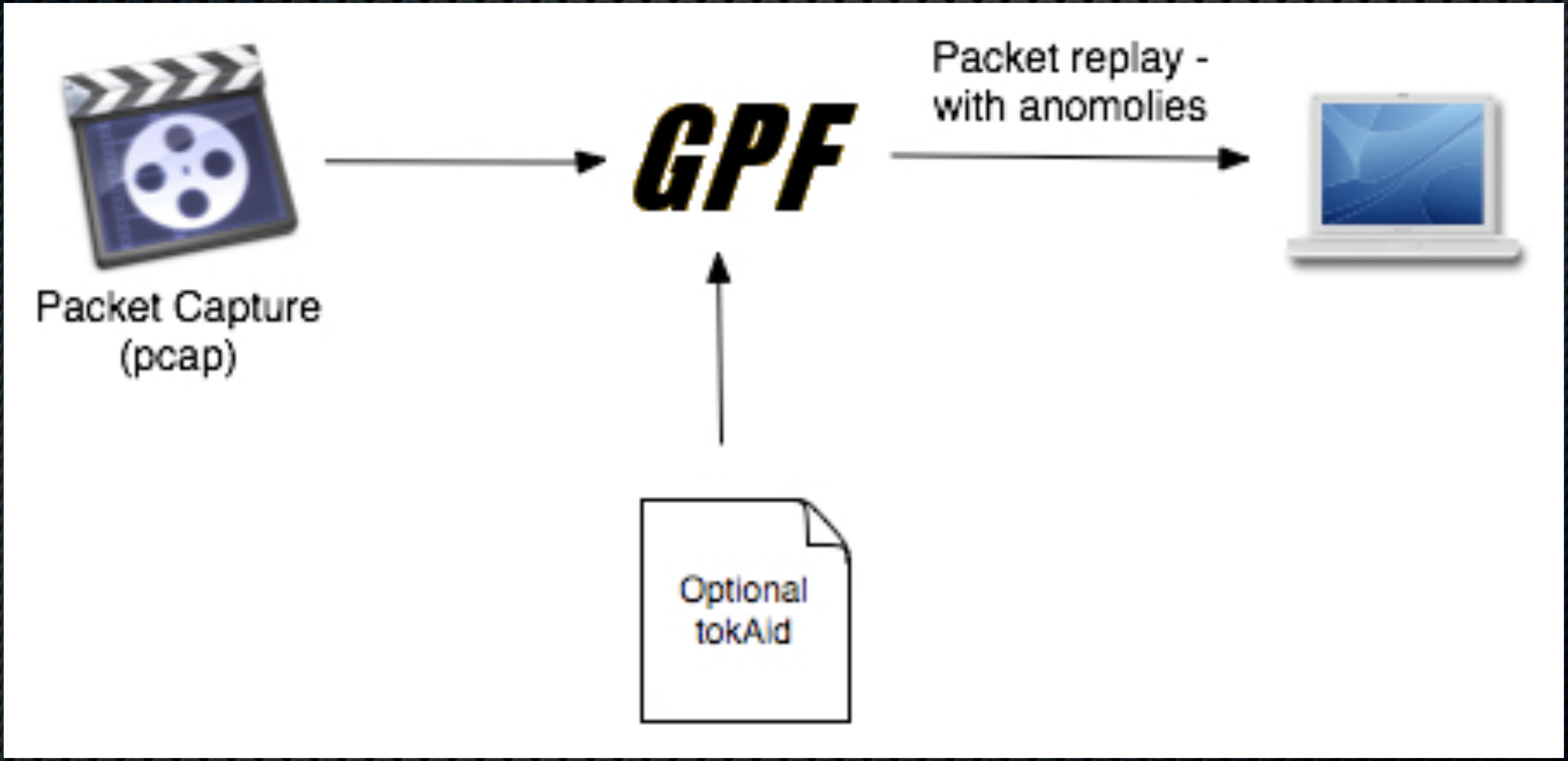
GPF

- Starts from a packet capture
- Written by the handsome and intelligent Jared DeMott
- Custom written “tokAids” describe the format of the packets, i.e. length fields, data type, etc.
  - Or default “ASCII”, “Binary” tokAid
- Randomly injects anomalies into the packets (according to the tokAid) and replays them repeatedly

Excerpt from mDNS tokAid

```
...
            tok=Create_Next_Tok(tok, leg);
            tok->type=LEN;
            tok->covered=1;
            tok->dataLen=1;
            Slurp_Into_Tok(tok, data);
            //Then the next token will be the string that len is associated with
            tok=Create_Next_Tok(tok, leg);
            tok->type=ASCII;
            tok->dataLen=_ndata_to_size8(tok->prev->data);
            Slurp_Into_Tok(tok, data);
            //check to see if we're at the end of the dns name
            if ( *(data+(tok->currentTotal)) == 0x00)
            {
                //the null is it's own token
                tok=Create_Next_Tok(tok, leg);
                tok->type=BINARY_END;
                tok->dataLen=1;
                Slurp_Into_Tok(tok, data);
...
```

GPF


Packet Capture (pcap)
GPF
Packet replay - with anomolies
Optional tokAid

Sulley

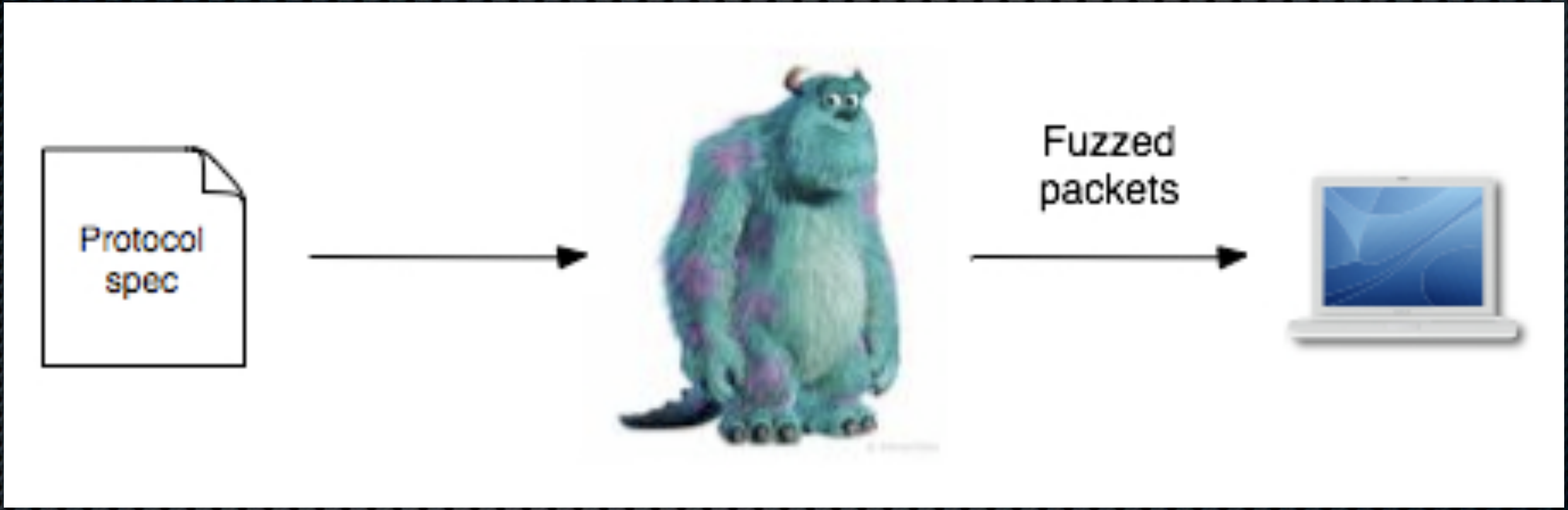
- A fuzzing framework
- User supplies a protocol description to the framework
- Framework systematically changes each described field to a set of anomalies
- No randomness, each test case tests something different
- Finite run time

Excerpt from mDNS Sulley File

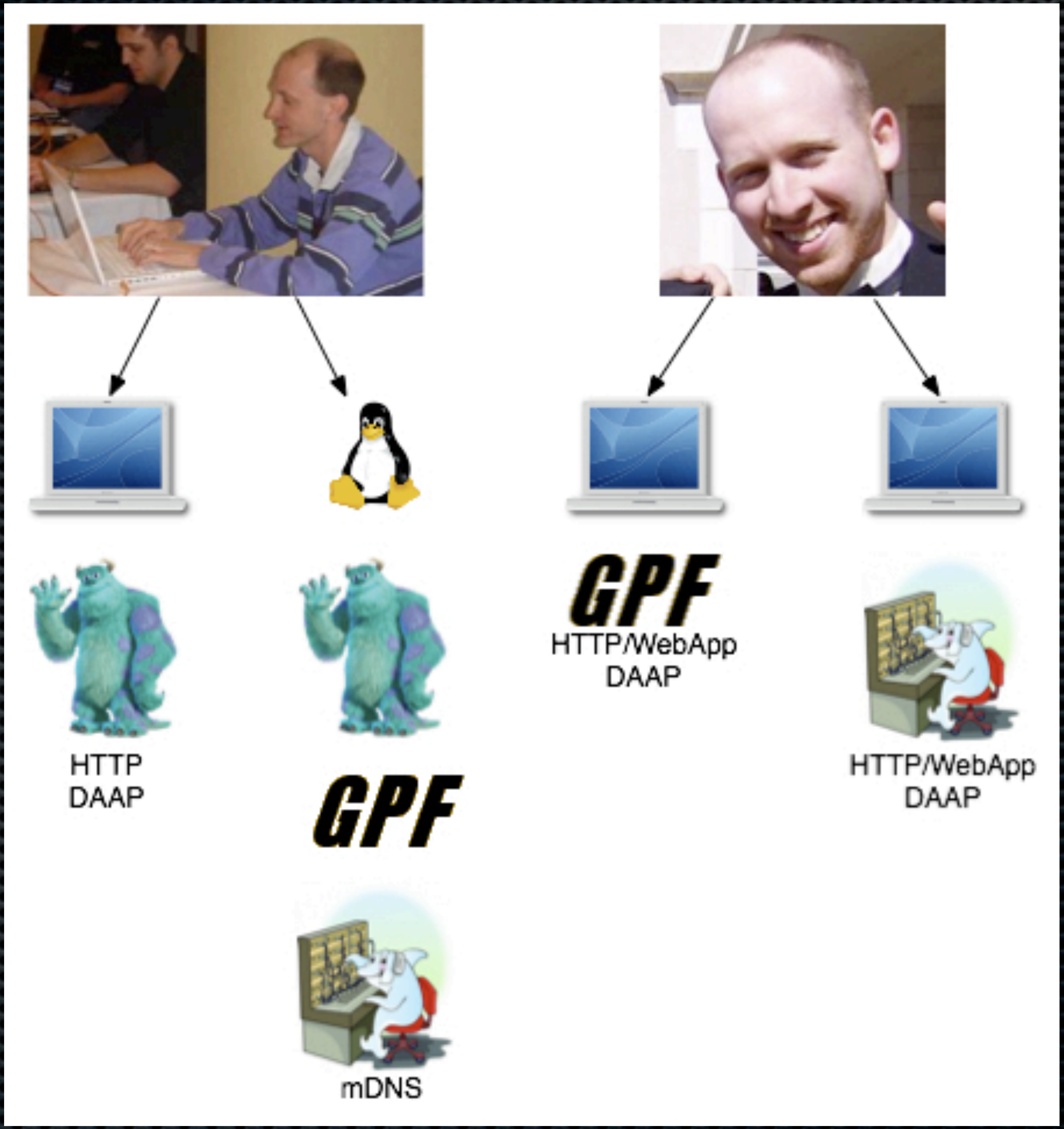
```
if s_block_start("query"):
        if s_block_start("name_chunk"):
                s_size("string", length=1)
                if s_block_start("string"):
                        s_string("A"*10)
                s_block_end()
        s_block_end()
        s_repeat("name_chunk", min_reps=2, max_reps=40, step=2, fuzzable=True, name="aName")

        s_group("end", values=["\x00", "\xc0\xb0"])   # very limited pointer fuzzing
        s_word(0xc, name="Type", endian='>')
        s_word(0x8001, name="Class", endian='>')
s_block_end()
s_repeat("query", 0, 1000, 40, name="queries")
```

Sulley



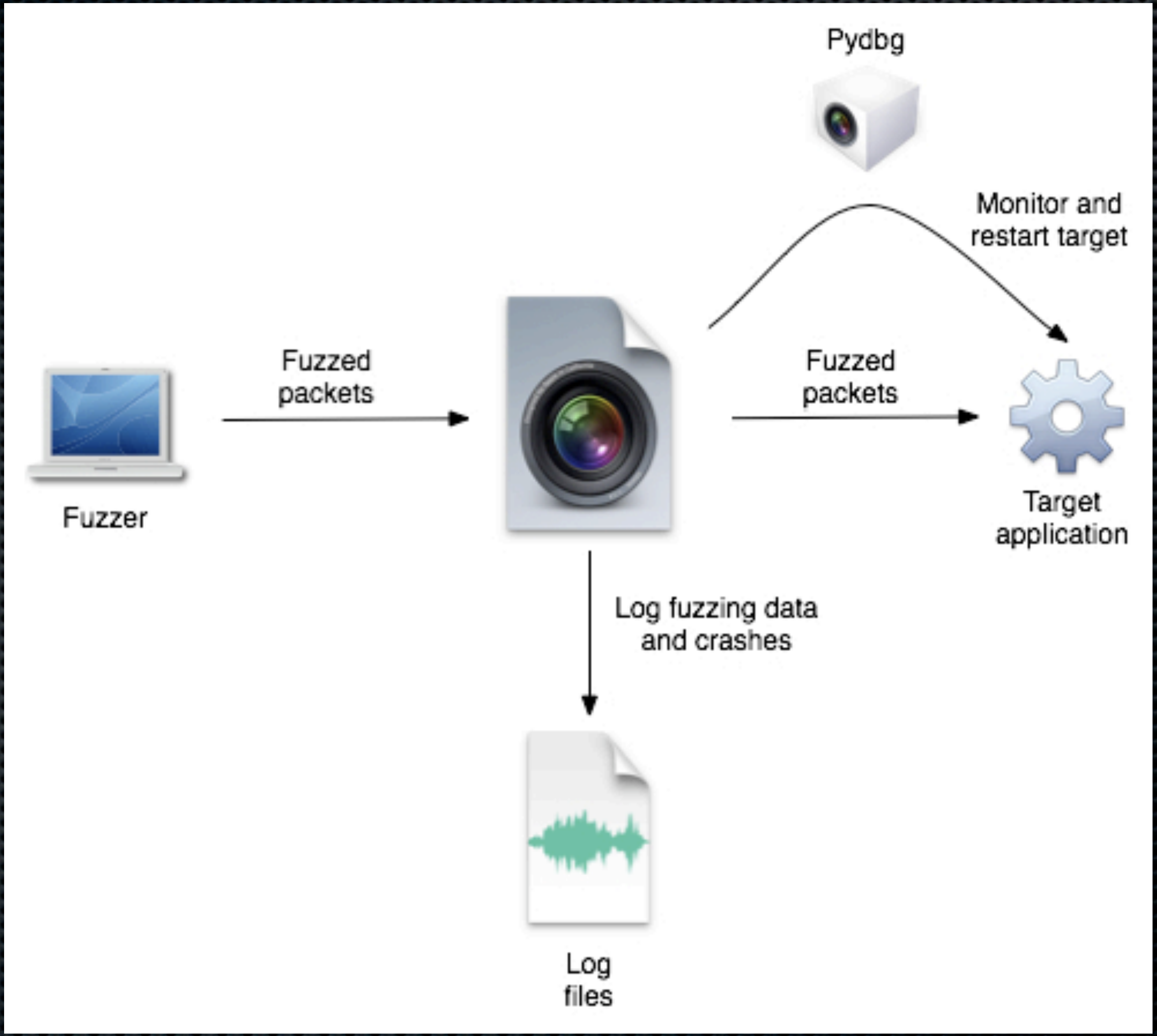
The Plan



Monitoring on Mac: MothaFuzza Monita

- Transparent Python proxy
- Records fuzzed data and responses
- Attaches to target and monitors health (with Pydbg)
- Logs crash reports and restarts target
- Can repeat captured data to help in crash analysis
- ***Works independent of the fuzzer being used***

MothaFuzza Monita



Iron Chef is Hard

- Target has a large attack surface
  - HTTP, DAAP, web application, mDNS, at least
- 60 minutes minus build and setup time (x4 machines)
- In real life, we'd probably fuzz this for a day or two per protocol per fuzzer (a week or two)
- You saw all the hard parts, just not the "sit back and wait for bugs" part

Sulley Didn’t finish

- Each Sulley test case more or less independent
  - Can’t skip any without possibly missing bugs
- Sulley DAAP fuzzer has 26,283 test cases
- Sulley standard HTTP fuzzer has 58,493 test cases
- Sulley normally does 1 test case per second
- Can be sped up, but can’t do 85k in an hour
- In real life, this isn’t an issue: “Run it and forget it”

With more time...

- Would customize test cases
  - i.e. “dialect” of the protocol
  - (which HTTP headers, variables, etc)

I want more time!

- Not enough time to analyze and redo fuzzing
    - Got code coverage but couldn’t make and send new test cases to expand coverage

What we *didn't* test

Bug(s)