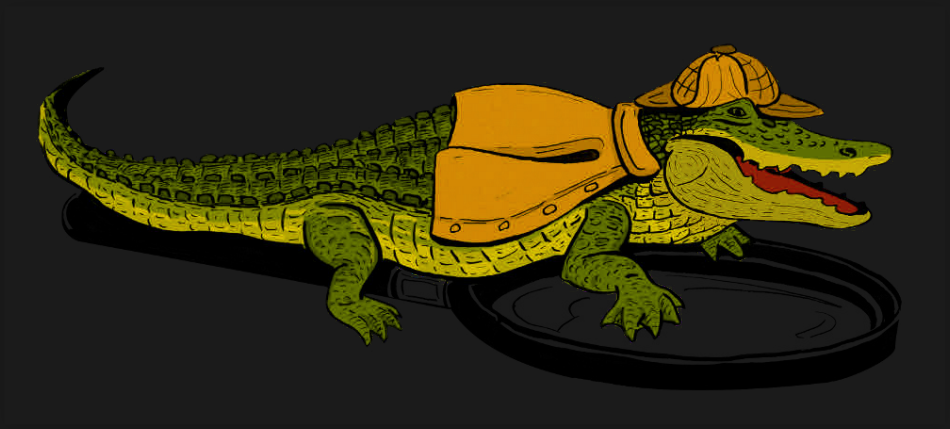
MIG
Mozilla
InvestiGator



Distributed and Real-Time Digital Forensics at the Speed of the Cloud

Julien Vehent / @jvehent

This presentation is online at https://goo.gl/OZ9ksE

Julien Vehent, Operations Security

- we respond to security investigations and incidents
- we help developers design and implement security controls
- we build tools & services to keep Mozilla secure

A post-APT1 security world


```
<indicatoritem id="1f3aff31-1155-4003-968c-40e5bd11e46e" condition="i
    <context document="FileItem" search="FileItem/Md5sum" type="mir">
    <content type="md5">3ce55c6994101faec00b5b7c2fee494f</content>
</context></indicatoritem>
```

That unfortunate commit...

git commit -a . && git push github master

3 lines (3 sloc) | 0.119 kb — Raw | Blame | History

```
1 [Credentials]
2 aws_access_key_id = AKIA[redacted]
3 aws_secret_access_key = [redacted]ev6
```

$ mig file -path / -name "^\.boto$" -content "abcdef123456" -size "<1k" -maxdepth 5

We're building a better Internet

Strong startup/incubator mindset

- Experiment & fail fast
- Minimalistic centralization
- Everyone can write and host a website...
- ...sometimes using operational standards

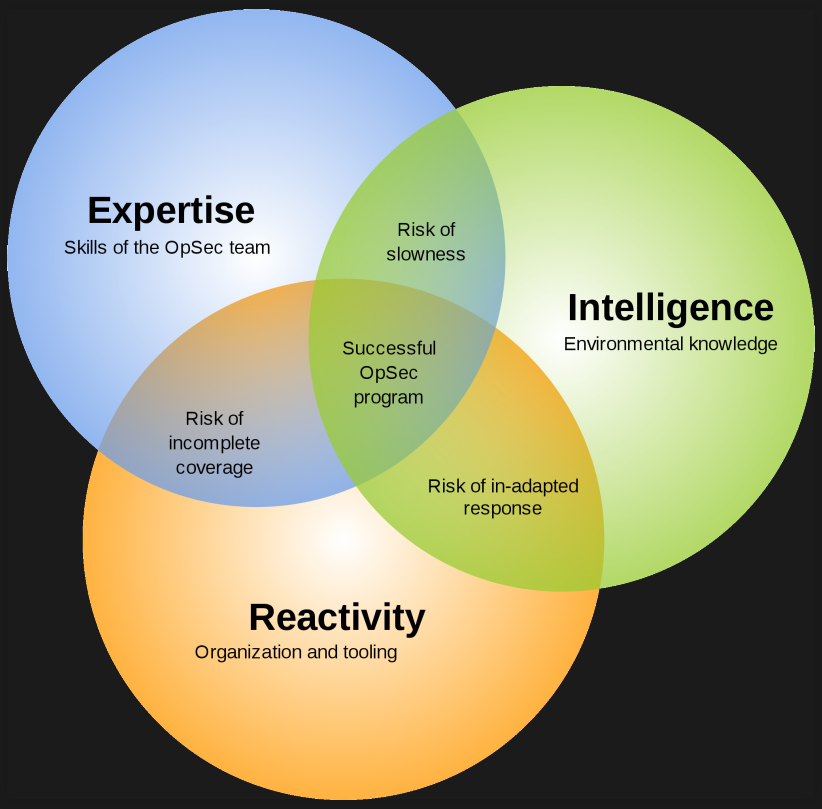
Security at the perimeter doesn't work when your infrastructure lives all over the internet

- 400+ active websites & services
- a dozen offices, hundreds of remotees
- 2 datacenters, tons of AWS accounts, heroku, rackspace, ...

all loosely connected only when needed

Incident Response at Mozilla

Need for a strong Operations Security group



3 OpSec problems

Visibility	too many systems doing too many things in too many ways. Need to see them all.
Reachability	we don't have accounts or network accesses to all systems.
Heterogeneity	every *snowflake* system is investigated in its own special way.

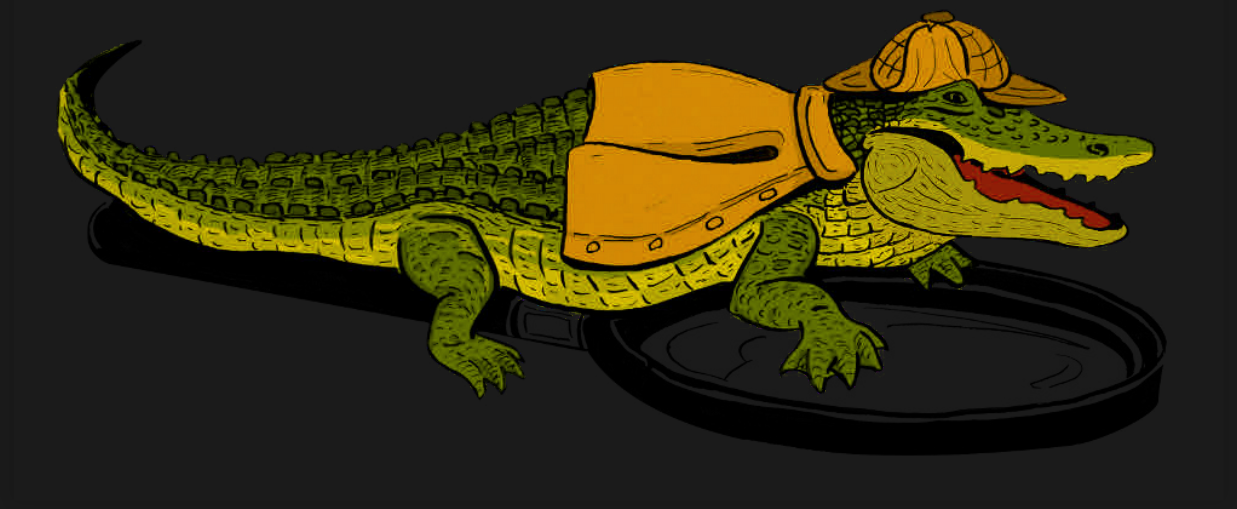
99% of investigations are simple

- "Hey Systems, seen that <file|ip|process|user> lately?".
- "Nope"
- "Nope"
- "Nope"
- "Oh yeah, just yesterday!" <--- OMG DEFCON 1!!!

The faster we run investigations, the more we will investigate.

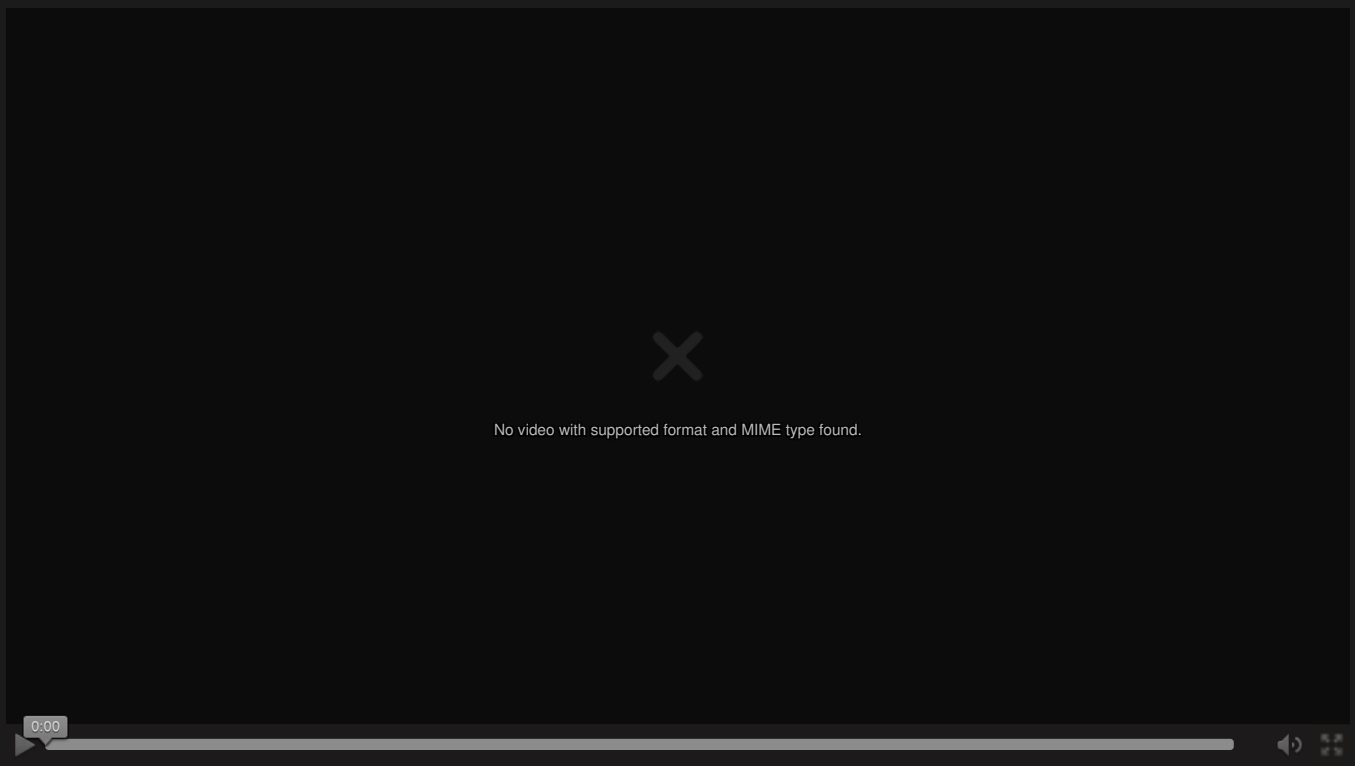
- bob left the company, did we revoke all his accesses?
- massive libstuff1 vulnerability, is it used anywhere?
- found IP 13.37.66.66 brute forcing the VPN, check other nodes to see if it's connected
- jean-kevin put some AWS key on pastebin, is it configured anywhere?
- anyone remembers that weird host that was running an anonymous proxy?

We couldn't find a tool we liked, so we built one

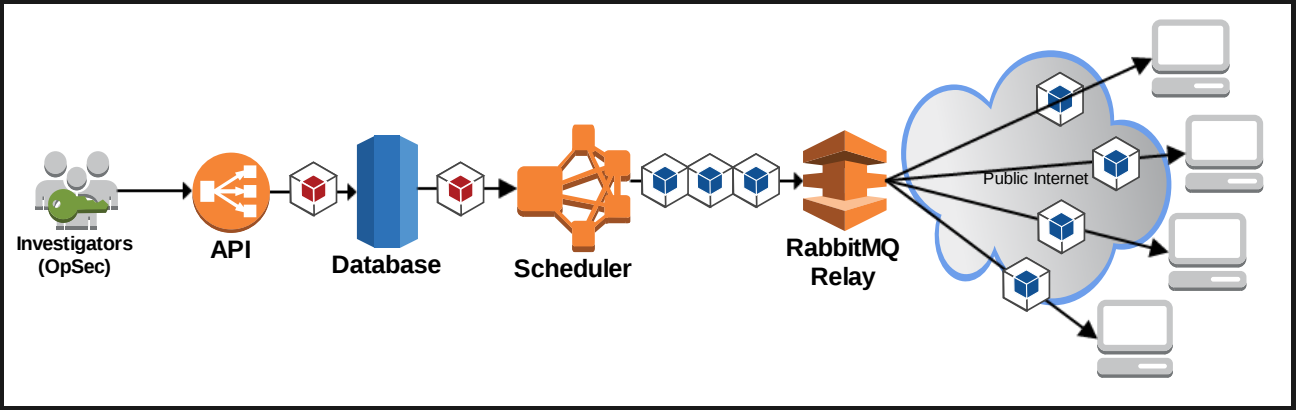


Demos!

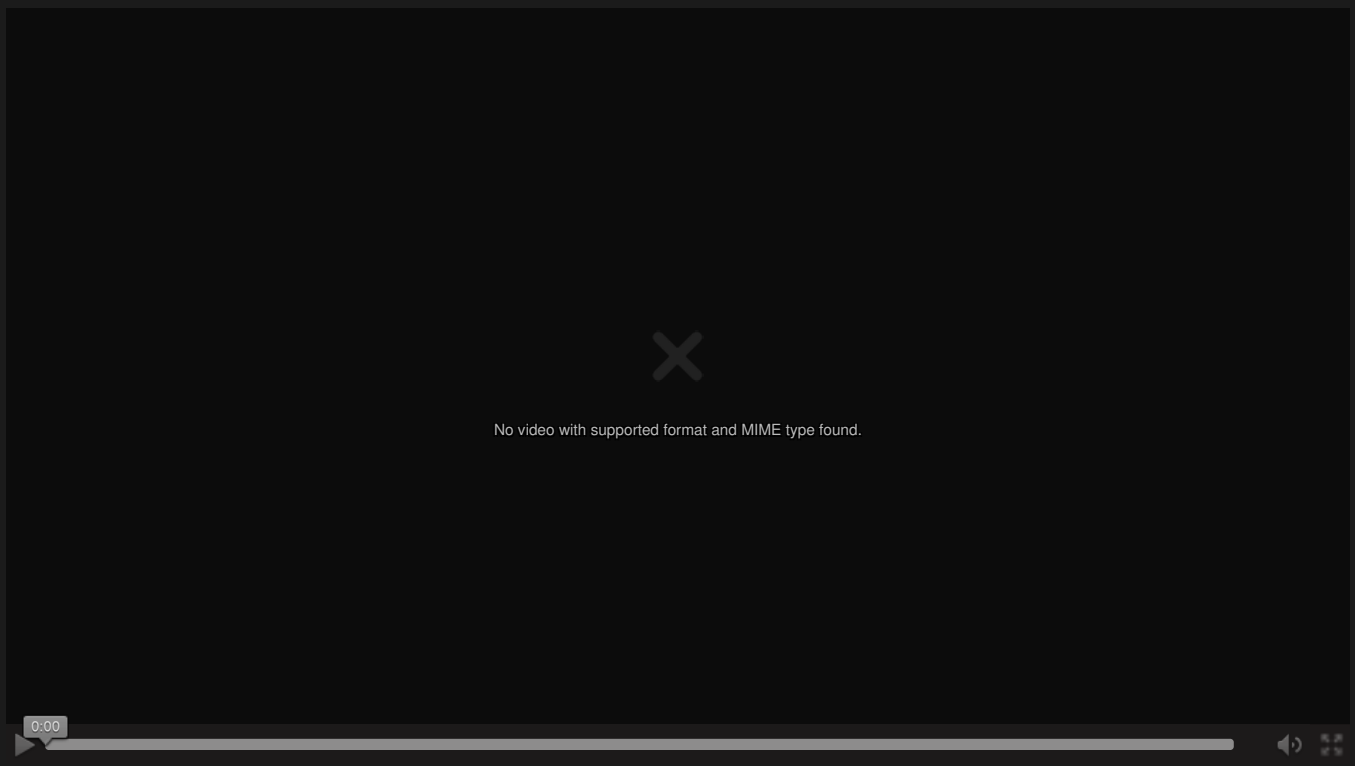
Locating a cron job that contains a password



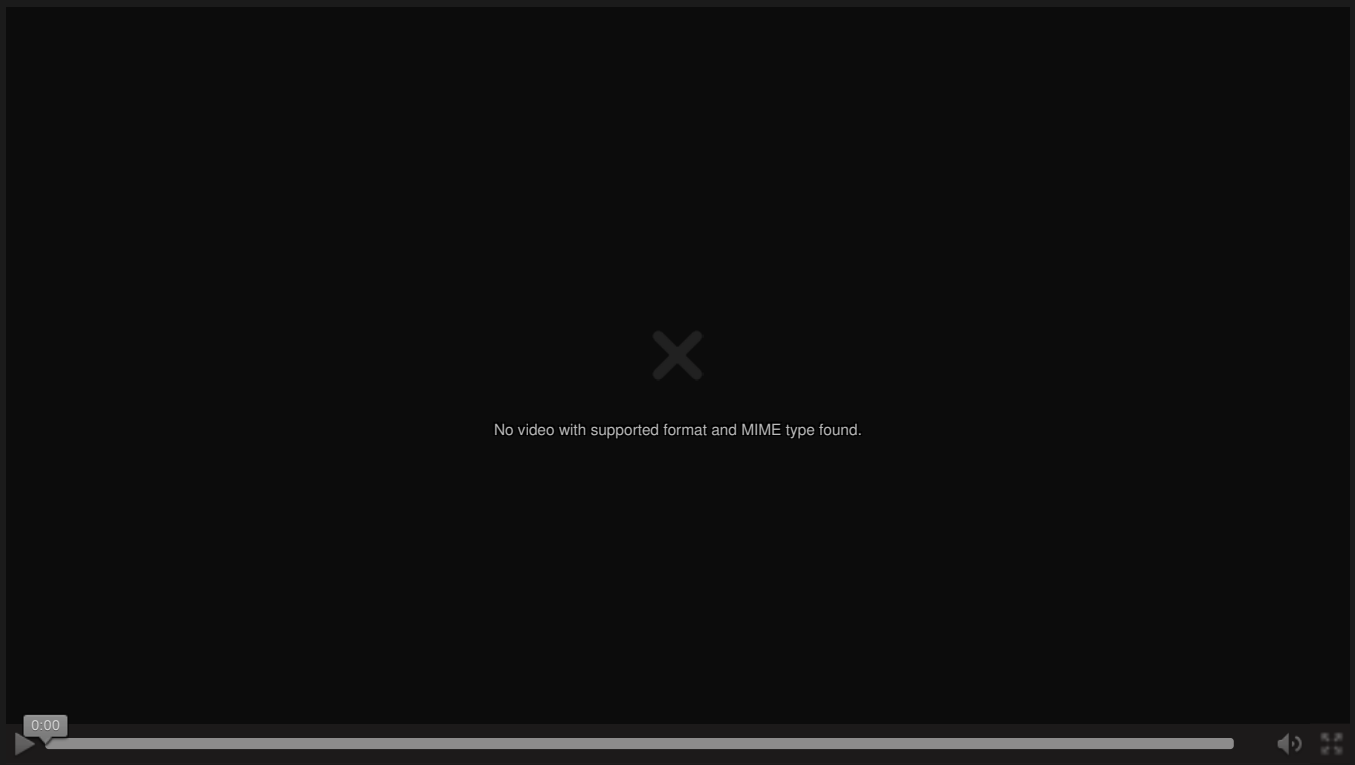
Basic investigation workflow



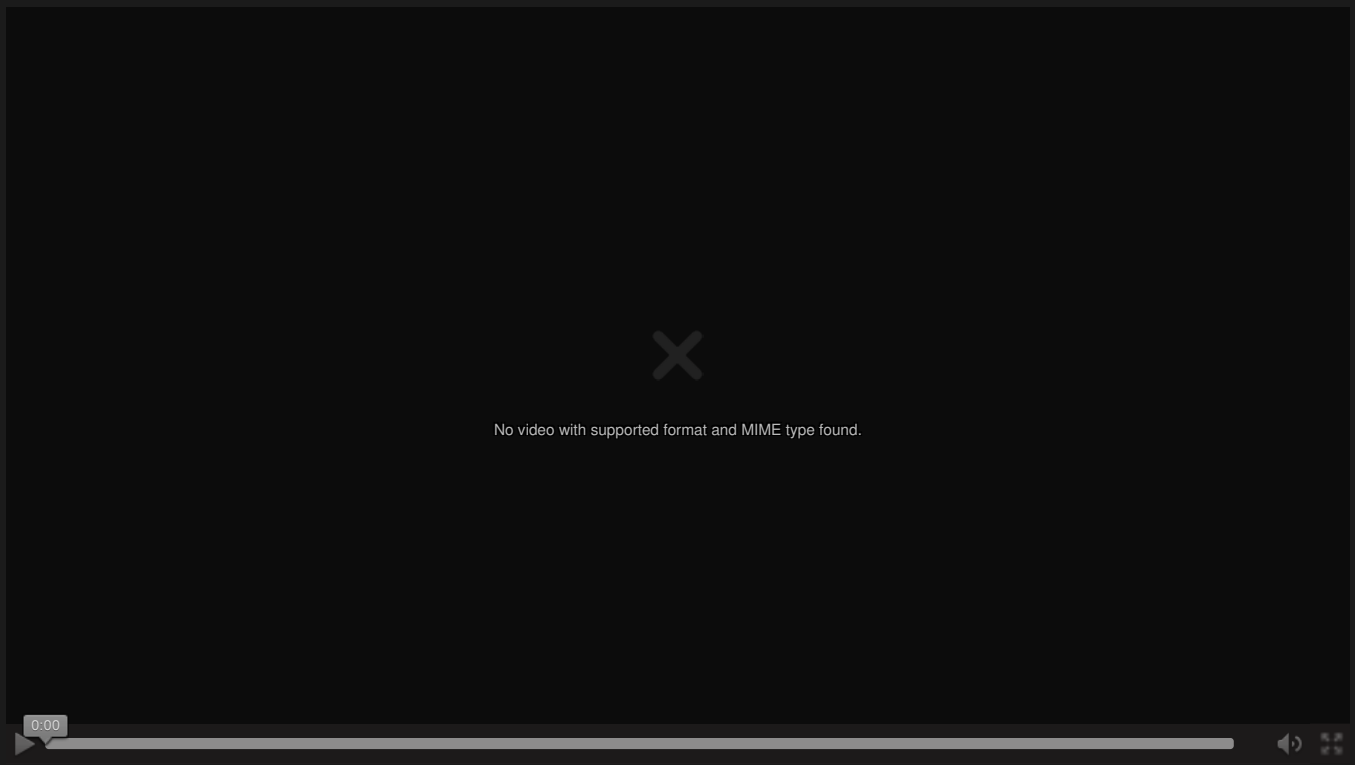
Got any private keys in those home folders?



Is that botnet IP connected anywhere?



Scan processes memories for a regex



Digital forensics on steroids

Massively Distributed means Fast.

Simple to deploy and Cross-Platform.

Secure! Don't trust until you verify.

Don't spy on data, respect Privacy.

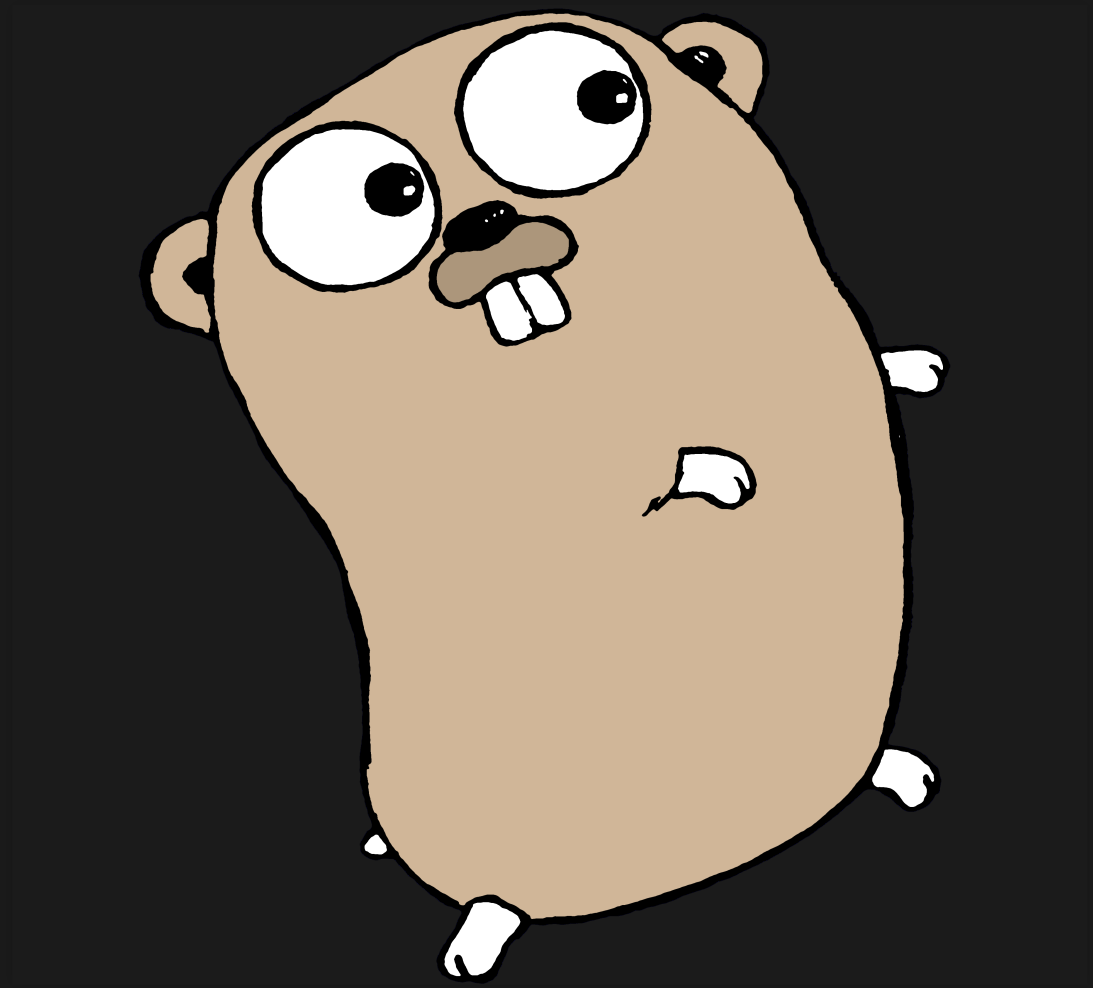
Massively distributed

Central platform only passes messages around.
Investigation & computation is done on the agents.

Small infrastructure footprint
means easier to deploy and operate.

Simple to deploy

One static binary, no dependencies.
Configuration is built-in or deployed via provisioning.



As secure as PGP, so pretty good


```
{
  "name": "locate bad actor",
  "target": "mode='daemon'",
  "validfrom": "2015-05-27T00:29:29.038012Z",
  "expireafter": "2015-05-27T00:30:59.038012Z",
  "operations": [
    { "module": "file",
      "parameters": {
        "searches": { "s1": { "paths":      ["/etc/cron.d"],
                              "contents":   ["badpassword"]
    }}}}
  ],
  "pgpsignatures": ["wsBcBAABCAAQBQJVZRAlCRCj1lIXO3..."],
  "syntaxversion": 2
}
```

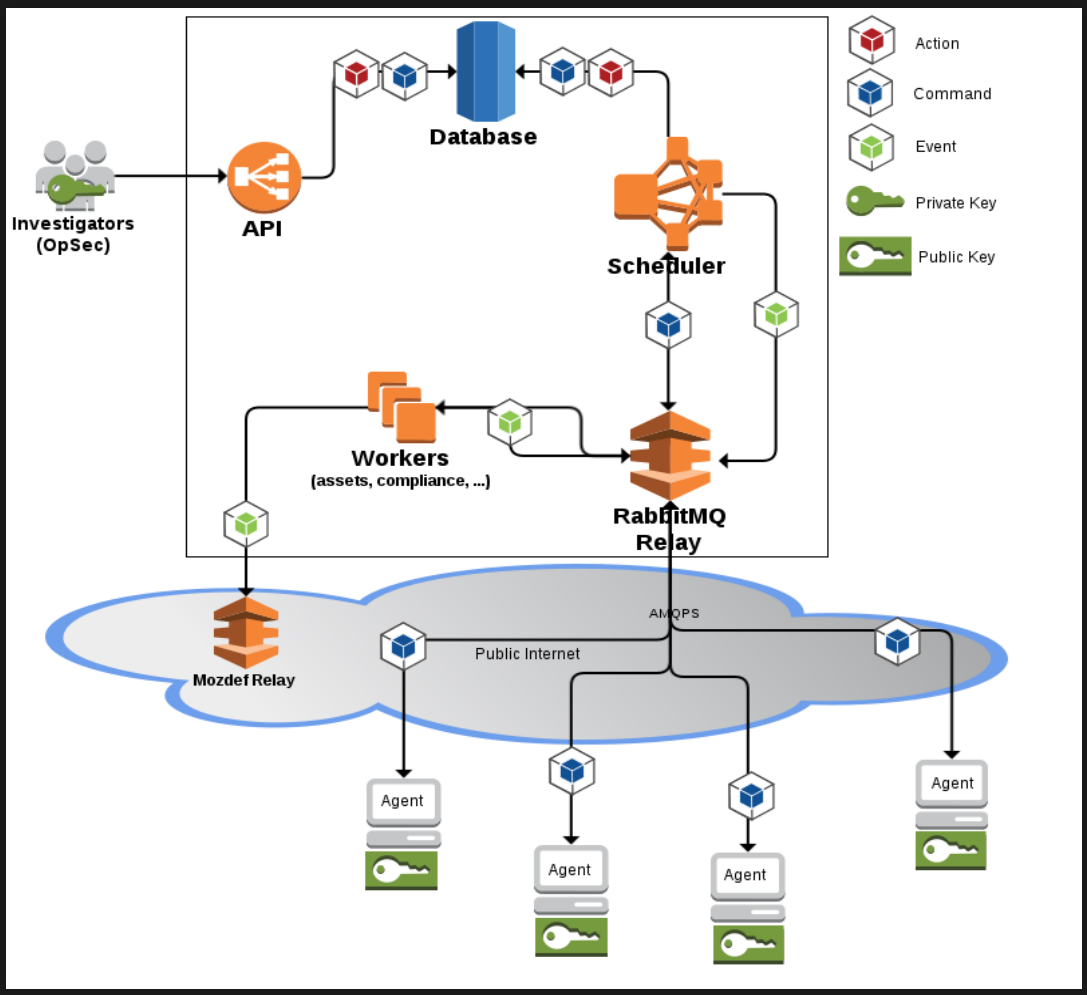
Privacy: I used to be a spy...

Doing forensics without full access to all data is hard, but not impossible.

Agents do the work and return answers without raw data.

Internals

REST API receives signed JSON messages
distributed to agents via RabbitMQ
and stored in a Postgres database.


Action
Command
Event
Private Key
Public Key
Database
Investigators
(OpSec)
API
Scheduler
Workers
(assets, compliance, ...)
RabbitMQ
Relay
Mozdef Relay
AMQPS
Public Internet
Agent
Agent
Agent
Agent

Architecture of the Agent

Security of the Agent

Agent only runs something if these conditions are met:

1. action has valid PGP signatures
2. issued by trusted investigators
3. with ACL accesses to a given module

multiple signatures required to run sensitive modules

Agent ACLs

```
var AGENTACL = [...]string{
`{ "file": { "minimumweight": 2,
    "investigators": {
        "Alice": { "fingerprint": "E60892BB..", "weight": 2 },
        "Bob": {   "fingerprint": "AD595634..", "weight": 2 } } }
}`,
`{ "memory": { "minimumweight": 3,
    "investigators": {
        "Alice": { "fingerprint": "E60892BB..", "weight": 2 },
        "Eve": {   "fingerprint": "F6D781AE..", "weight": 1 } } } }
}`,
}
```

Agent ACLs

The weights of each investigator providing a valid signature are summed, and if the total weight is equal or higher than the minimum weight, the operation is considered valid.

```
TotalWeight = Weight[Alice} + Weight[Bob]
if TotalWeight >= MinimumWeight { run module }
```

Security of the platform: API

REST API, uses IdFix PGP Token authentication

```
curl -H 'X-PGPAUTHORIZATION: 1;2015-05-28T15:04:05Z;111;owEBYQGe/pANAwAIAaP
https://api.mig.example.net/api/v1/
```

PGP already needed to sign actions

A PGP Token avoids needing another username/password.

Security of the platform: Database

Typical PostgreSQL protections (TLS, credentials, GRANTs)

Minimalistic attack surface:

- Investigator keys are not stored in DB
- Results are in DB but never contain raw data, minimize impact of leak

Security of the platform: Scheduler

Complex code path to move messages around BUT:

- No user interaction
- No way to tamper with signatures

Security of the platform:

Most exposed component (public).
Requires AMQP over TLS with client certs and credentials.
Tightly controlled RabbitMQ ACLs, but hard to write/audit.

Complex investigations: write JSON directly

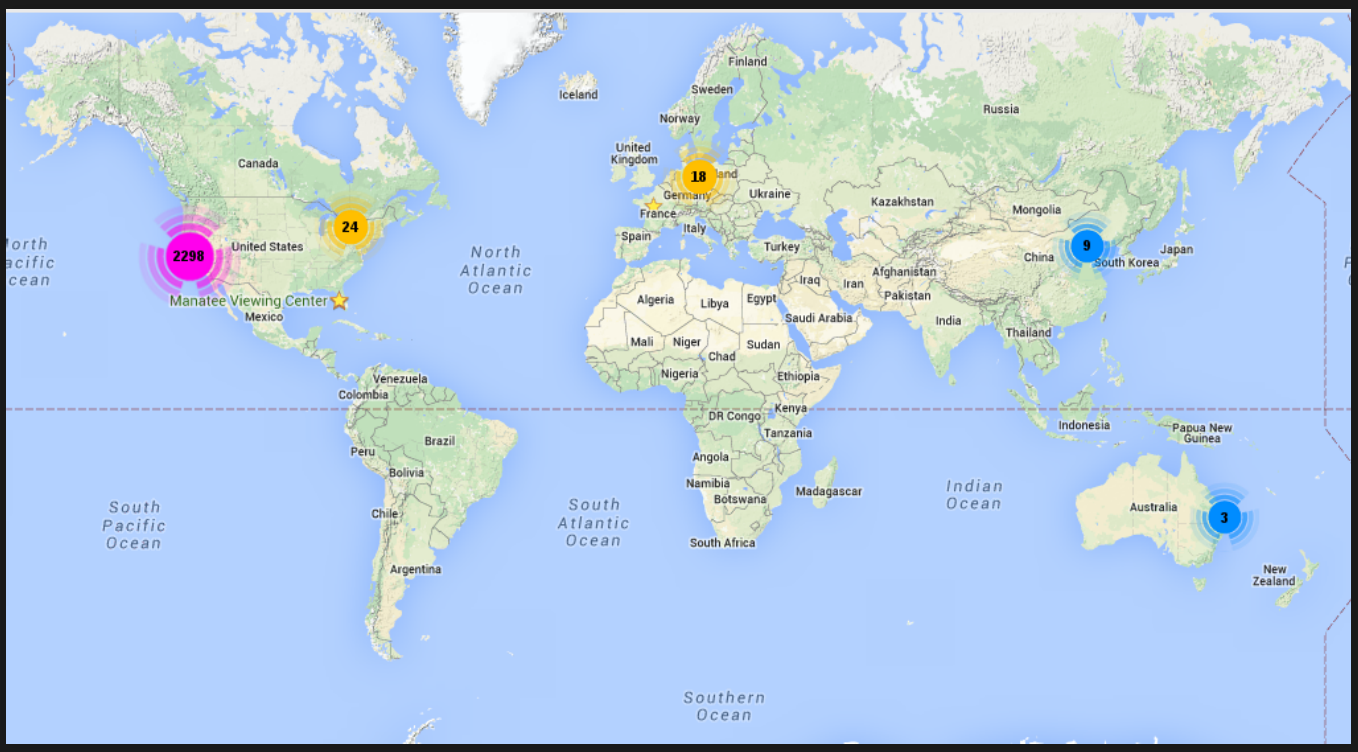
many samples at
https://github.com/mozilla/mig/tree/master/actions

Example: Shellshock IOCs

```
{
  "name": "Shellshock IOCs (nginx and more)",
  "target": "environment->>'os' IN ('linux','darwin') AND mode='daemo
  "operations": [
    {
      "module": "file",
      "parameters": {
        "searches": {
          "iocs": {
            "paths": [
              "/usr/bin",
              "/usr/sbin",
              "/bin",
              "/sbin",
              "/tmp",
              "/var/tmp"
            ],
            "sha256": [
              "73b0d95541c84965fa42c3e257bb349957b3be626dec9d55efcc6e
              "ae3b4f296957ee0a208003569647f04e585775be1f3992921af996
              "2d3e0be24ef668b85ed48e81ebb50dce50612fb8dce96879f80306
              "2ff32fcfee5088b14ce6c96ccb47315d7172135b999767296682c3
              "1f5f14853819800e740d43c4919cc0cbb889d182cc213b0954251e
              "2bc9a2f7374308d9bb97b8d116177d53eaca060b562f6f66f5dd1a
            ],
            "contents": [
              "/bin/busybox;echo -e '\\\\147\\\\141\\\\171\\\\146\\\\
              "legend.rocks"
            ],
            "names": [
              "legend.txt"
            ]
          }
        }
      }
    },
    {
      "module": "netstat",
      "parameters": {
        "connectedip": [
          "108.162.197.26",
          "162.253.66.76",
          "89.238.150.154",
          "198.46.135.194",
          "166.78.61.142",
          "23.235.43.31",
          "54.228.25.245",
          "23.235.43.21",
          "23.235.43.27",
          "198.58.106.99",
          "23.235.43.25",
          "23.235.43.23",
          "23.235.43.29",
          "108.174.50.137",
          "201.67.234.45",
          "128.199.216.68",
          "75.127.84.182",
          "82.118.242.223",
          "24.251.197.244",
```

```
          "166.78.61.142",
          "119.110.98.93",
          "2.0.1.5"
        ]
      }
    }
  ],
  "description": {
    "author": "Julien Vehent",
    "email": "ulfr@mozilla.com",
    "revision": 201410031030
  },
  "syntaxversion": 2
}
```

Visualizing results on a map



A generic security platform

- Security Compliance
- Vulnerability scanning with mozilla/mozoval
- Syscall auditing (auditd) with mozilla/audit-go
- Log inspections (OSSEC style)
- Network monitoring (distributed NSM) using GoPacket

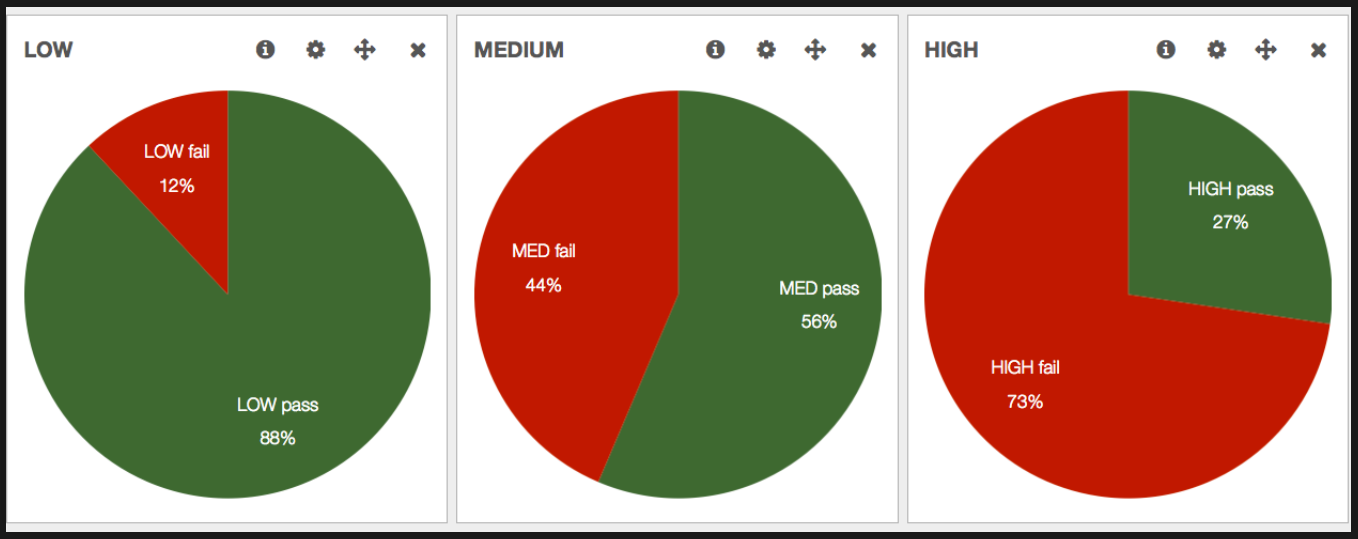
Measuring security compliance

```
{
    "module": "file",
    "parameters": {
        "searches": {
            "checkforverboselogging": {
                "paths": [
                    "/etc/ssh/sshd_config"
                ],
                "contents": [
                    "(?i)^loglevel verbose$"
                ]
            },
            "checkpasswordusageisoff": {
                "paths": [
                    "/etc/ssh/sshd_config"
                ],
```

Storing security compliance

```
{
    "name":          "checkpasswordusageisoff",
    "description":   "compliance check for ssh",
    "target":        "server1.mydomain.example.net",
    "utctimestamp": "2015-02-19T02:59:30.203004Z",
    "compliance":    true,
    "location":      "/etc/ssh/sshd_config",
    "ref":           "syslowremote1",
    "check": {
        "test": {
            "type":  "file",
            "value": "(?i)^passwordauthentication no$"
    }
    "tags": {
        "operator": "IT"
    },
```

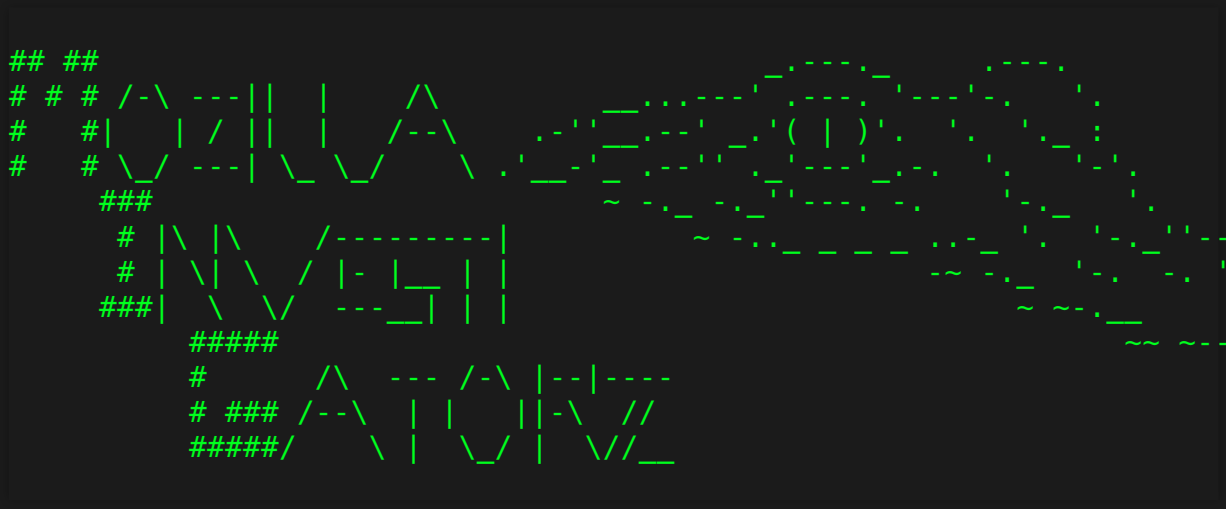
Graphing security compliance



tl;dr

MIG is made of distributed agents securely queried from a central platform to investigate the state of large pools of systems remotely.

Questions?



http://mig.mozilla.org