Mirror on the wall

Using Blue Team techniques in Red Team ops

BruCON 2018

Mark Bergman &
Marc Smeets

HACKING FOR B33R
BRUCON
WWW.BRUCON.ORG

OUTFLANK
clear advice with a hacker mindset

ABOUT US

Mark Bergman - @xychix

- Started in mainframe world in 1999, not the average developer. Moved to offensive security in 2004.
- Red Team operator and infra builder, repeat == python code

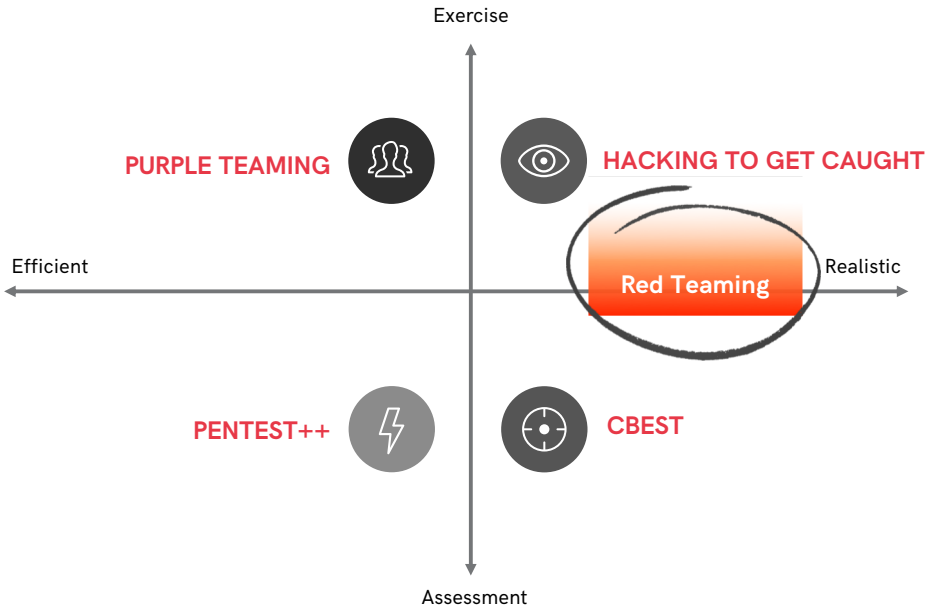
Marc Smeets - @mramsmeets

- In offensive security since 2006, background in system and network engineering
- Red Team operator and tool builder

Outflank

- Highly specialised in Red Teaming and attack simulation
- Outflank.nl/blog & github.com/OutflankNL


HAFFEN
SPORT


Exercise
PURPLE TEAMING
HACKING TO GET CAUGHT
Efficient
Red Teaming
Realistic
PENTEST++
CBEST
Assessment

MAIN QUESTIONS FOR US

1. **Can we become better in control of our infrastructure?**

2. **Can we detect analyses and detection by Blue?**

3. **Can we use that knowledge to improve and prolong the exercise?**

To answer we need to cover 3 main topics:

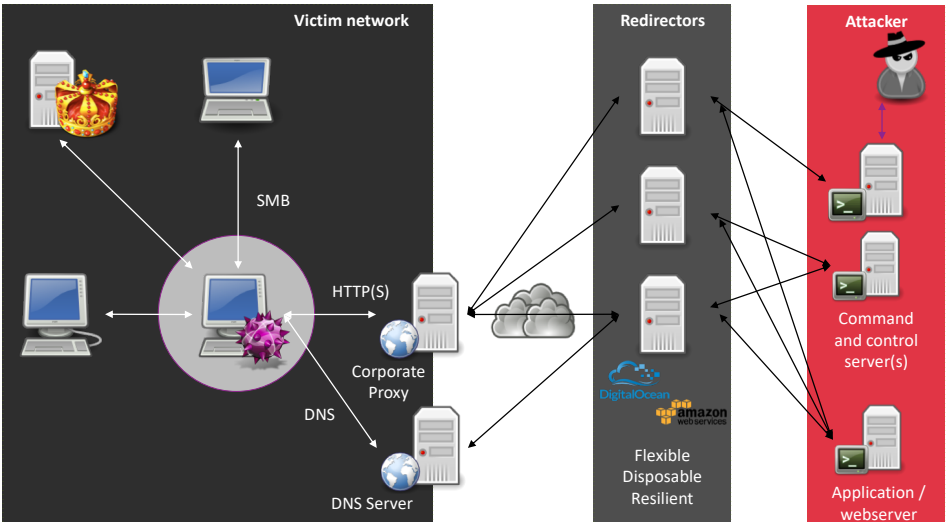
- Understanding of (advanced) red teaming infrastructures
- Monitoring relevant information from RT traffic data and RT ops
- How to detect analyses and detection, and use to our advantage?

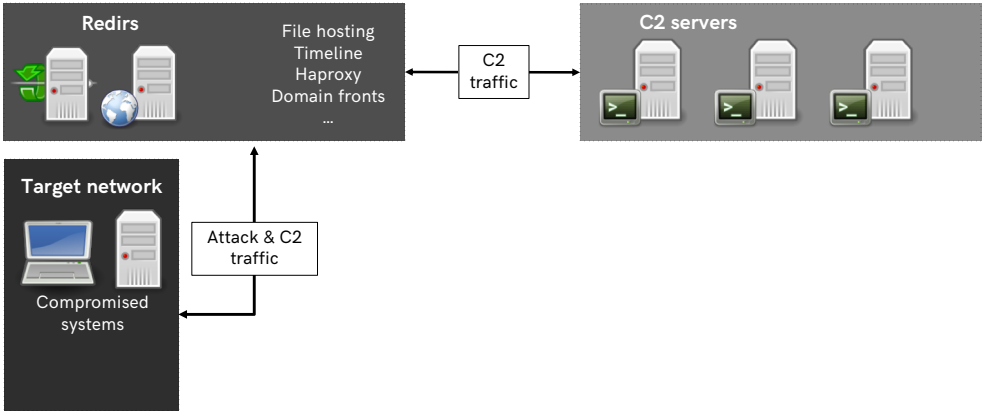
RED TEAMING INFRASTRUCTURES OVERVIEW

- Typical components every team uses, amongst others:
  - C2 team servers
  - Redirectors
  - Domain fronting
  - Throw away identities (email and LinkedIn)
  - VPN

- Components we use that are not really publicly documented, amongst others:
  - Multifunctional redirectors (multiple endpoints)
  - Web based file servers for HTML-smuggling
  - Tracking-pixel for timelines

COMMAND & CONTROL: INFRASTRUCTURE




Redirs
File hosting
Timeline
Haproxy
Domain fronts
...
C2
traffic
C2 servers
Target network
Attack & C2
traffic
Compromised
systems

OPERATIONAL CHALLENGES

Simply loosing oversight in the ops

- **Examples:**
  - Manual grepping in logs
  - No correlations between scenarios
  - No easy to access central repository (of keystrokes, screenshots, beacon logs)

CENTRALISING RED TEAM LOGS

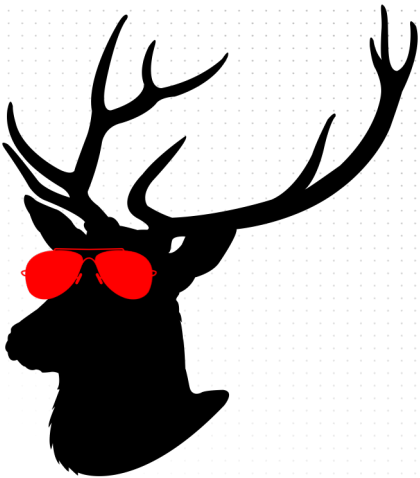
We have got:

- Traffic logs at many redirectors
- Red Team operational logs at many teamservers
- Relevant information in all those logs

We want situational awareness

- Easy viewing and historical searching of the entire operation
- Enrichment of our logs to make them extra useful
- Real-time dashboards, for us and for the White Team
- Not talking about system logs, e.g. SSH auth, sudo, firewall, etc.

This sounds like a problem Blue Teams have encountered long ago



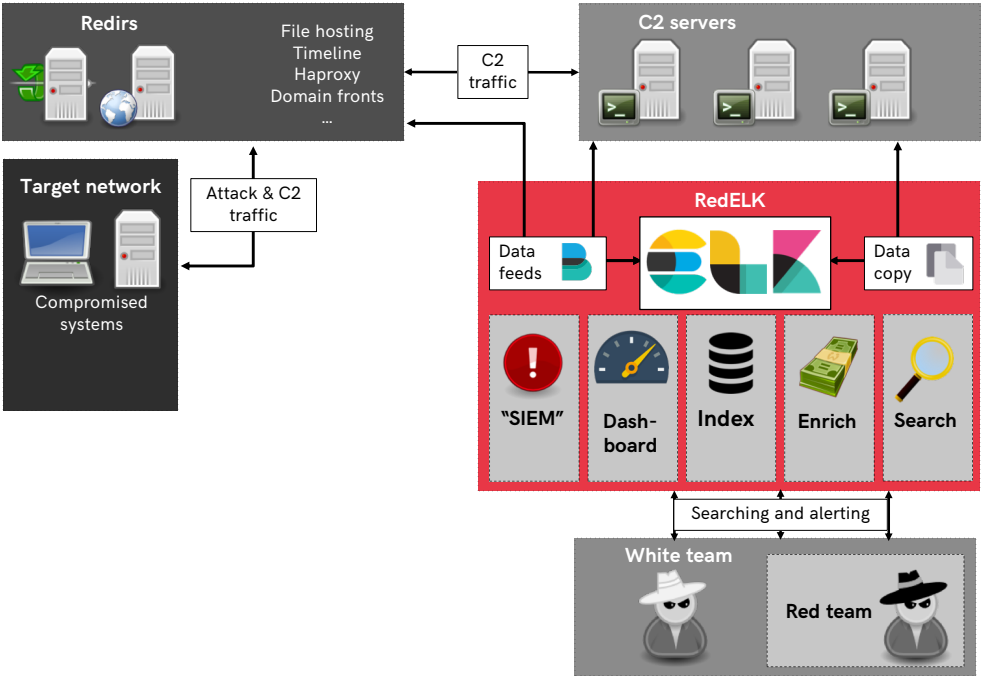
INTRODUCING 'RedELK'

RedELK

- Red Team's SIEM, based on the ELK stack
- Operational data as well as IOC data
- 1 central location of data for better situational awareness
- Automated installation to support disposable RT infrastructures

Current status

- Log aggregation of Cobalt Strike and HAProxy. Working on Apache.
- Fine grained filtering allows for very detailed querying
- Log enrichment: filter out test beacons, tag known good/bad, GeoIP, etc.
- Easy querying of screenshots, keystrokes, IOCs, operator logs, traffic, etc.
- Ready made smart dashboards, aggregations and queries
- Query for and alarm about Blue Team's defensive movements


Redirs
File hosting
Timeline
Haproxy
Domain fronts
...
C2 traffic
C2 servers
Target network
Attack & C2 traffic
Compromised systems
RedELK
Data feeds
Data copy
"SIEM"
Dash-board
Index
Enrich
Search
Searching and alerting
White team
Red team

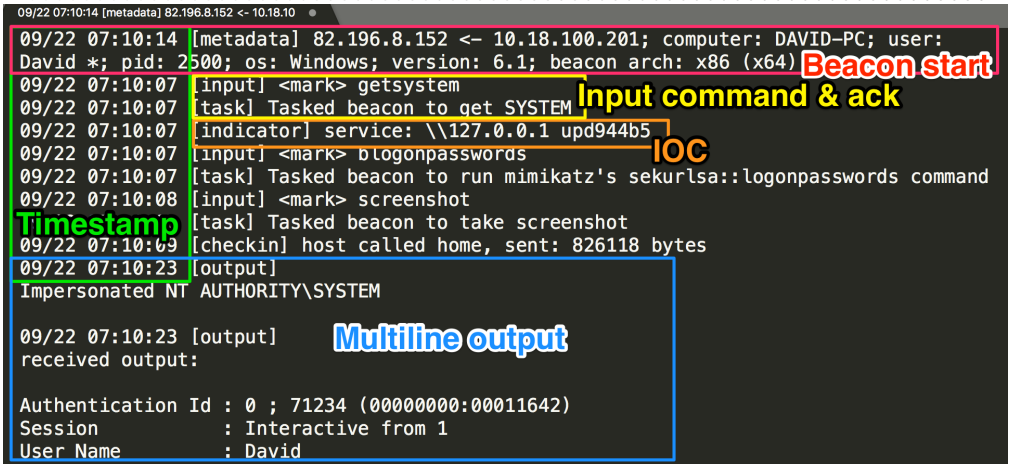
UNDERSTANDING COBALT STRIKE LOGS

Logs resides on teamserver at "cobaltstrike/logs/$DATE/"

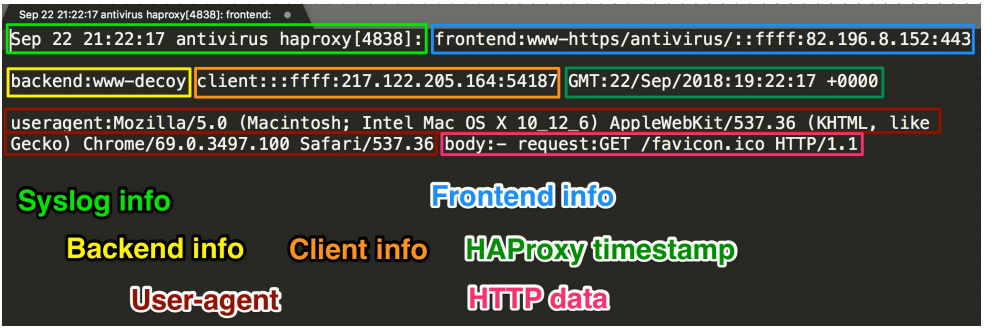
Type of data	Cobalt Strike log file name	Relevant?
Beacon transcripts	$IP/beacon_$BeaconID.log	Yes
Target screenshots	$IP/screenshots/screenshot_ $TIME_$BeaconID.jpg	Yes
Keylogger output	$IP/keystrokes/keystrokes_ $BeaconID.txt	Yes
Event log and operator chat	events.log	Somewhat
Screenshots of CS windows	screenshots/$OperatorName/$Time_*.png	No
Failed beacons	unknown/beacon_.log	No
Web server log	weblog.log	No
Downloads	downloads.log	No

We prefer default logging, no Aggressor scripts for modified logging

UNDERSTANDING BEACON LOGS



UNDERSTANDING HAPROXY LOGS



We use a custom HAProxy log format:

log-format frontend:%f/%H/%fi:%fp\ backend:%b\ client:%ci:%cp\ GMT:%T\
useragent:%[capture.req.hdr(1)]\ body:%[capture.req.hdr(0)]\ request:%r

At 'frontend' section:
declare capture request len 40000
http-request capture req.body id 0
capture request header User-Agent len 512

DATA ENRICHMENT

Cobalt Strike logs

Beacon metadata (1st line) info:

- target_user, target_ipint, etc

System classification:

- Tags: know_testsystem and known_sandbox

Usability improvement:

- Hyperlinks to full beaconlogs, keystrokes and screenshots
- Screenshot thumbnails

Redirector logs

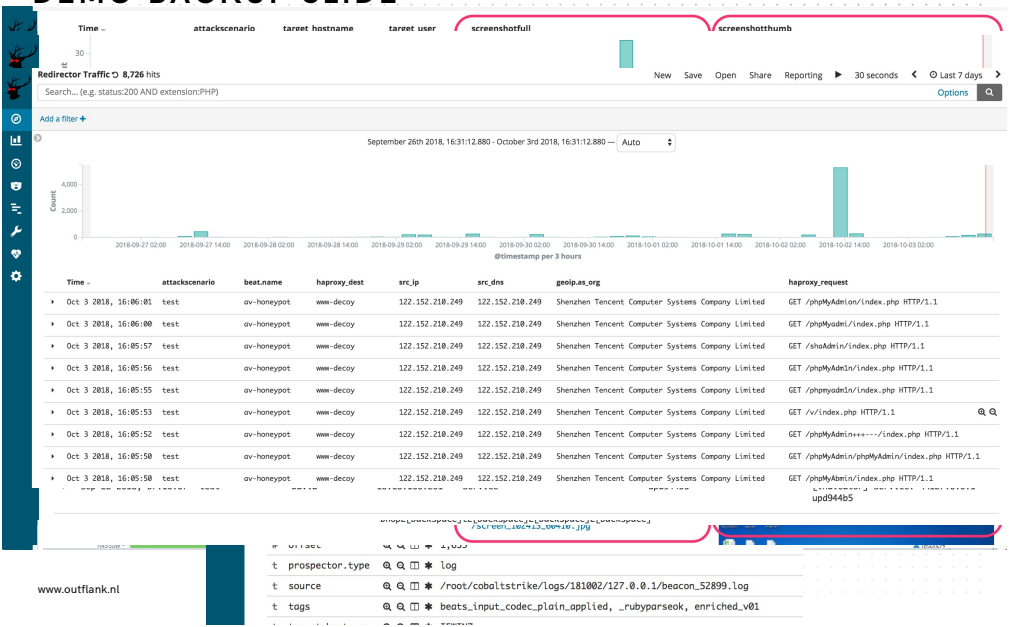
IP info:

- GeoIP, reverse DNS & WHOIS

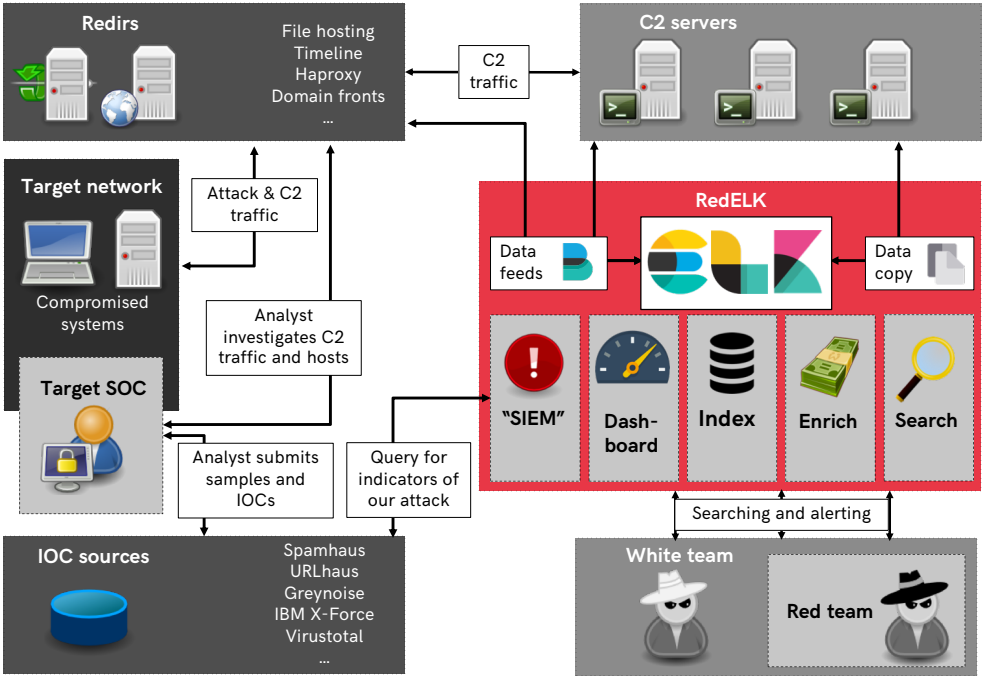
System classification

- Tags: torexitnode, iplist_customer, iplist_redteam, iplist_unknown
- Greynoise data

DEMO BACKUP SLIDE



www.outflank.nl


Redirs
File hosting
Timeline
Haproxy
Domain fronts
...
C2 traffic
C2 servers
Target network
Compromised systems
Attack & C2 traffic
RedELK
Data feeds
Data copy
Analyst investigates C2 traffic and hosts
Target SOC
"SIEM"
Dash-board
Index
Enrich
Search
Analyst submits samples and IOCs
Query for indicators of our attack
Searching and alerting
IOC sources
Spamhaus
URLhaus
Greynoise
IBM X-Force
Virustotal
...
White team
Red team

USING RedELK FOR DETECTION OF BLUE TEAM ACTIONS

Blueteam OPSEC mistakes:

- Uploading found samples to public services
  - Uploading artefacts to Virus Total
  - Testing payloads on sandboxes with alternative internet IPs

- Visiting the site or URL found in order to investigate
  - Manual visits with another user agent compared to beacon

DECOYS , IT'S LIKE A GAME

Now let's say we've learned enough from our first batch of logs to differentiate proper C2 traffic from other traffic.

- 'decoy' the investigators, give them something to chew on.

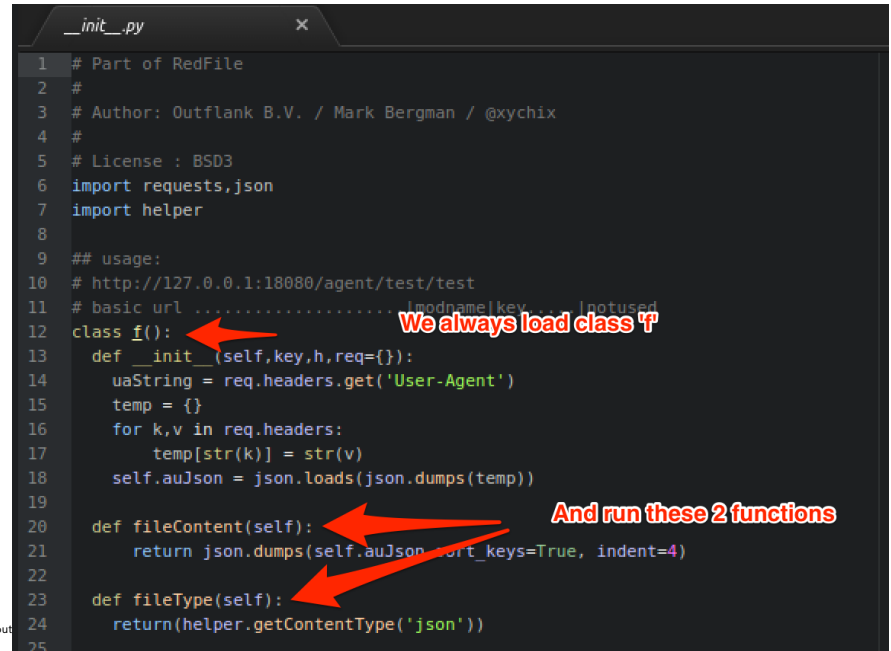
INTRODUCING 'RedFILE'

Serving files from code

- Basically every URL calls a python module which 'builds' the output.
- Base-code is 'thin' and accepts modules

Some ideas

- Return content based on user agent
- Return content only when a valid 'key' is present and a key can only be used 'n' times. Even more interesting is what we serve when the key is reused.
- Return content only N minutes after another call


__init__.py
Part of RedFile
#
Author: Outflank B.V. / Mark Bergman / @xychix
#
# License : BSD3
import requests,json
import helper
usage:
# http://127.0.0.1:18080/agent/test/test
# basic url ..... |mpdname|key|...|notused|
class f():
def __init__(self,key,h,req={}):
uaString = req.headers.get('User-Agent')
temp = {}
for k,v in req.headers:
temp[str(k)] = str(v)
self.auJson = json.loads(json.dumps(temp))
def fileContent(self):
return json.dumps(self.auJson, sort_keys=True, indent=4)
def fileType(self):
return(helper.getContentType('json'))
We always load class 'f'
And run these 2 functions

HOW DOES THIS IMPROVE RED TEAMING?

Blue often has to learn

- Looking at the right incidents and realize stuff might change.
- Ransomware often is offline quite fast after the hit, RedFile might help Blue to anticipate on this behaviour.

Will we be able to downplay an incident by offering valid but less threatening content?

"Targeted? Nah just a bitcoin stealer"

SUMMARY

Boxing instead of wrecking ball

RedELK

OPSEC for blue

https://outflank.nl/blog/

https://github.com/outflanknl

OUTFLANK

clear advice with a hacker mindset

Mark Bergman & Marc Smeets

mark@outflank.nl marc@outflank.nl