Canaries in a Coal Mine

Detecting Lateral Movement using the OpenCanary Honeypot

@petermorin123

Peter Morin

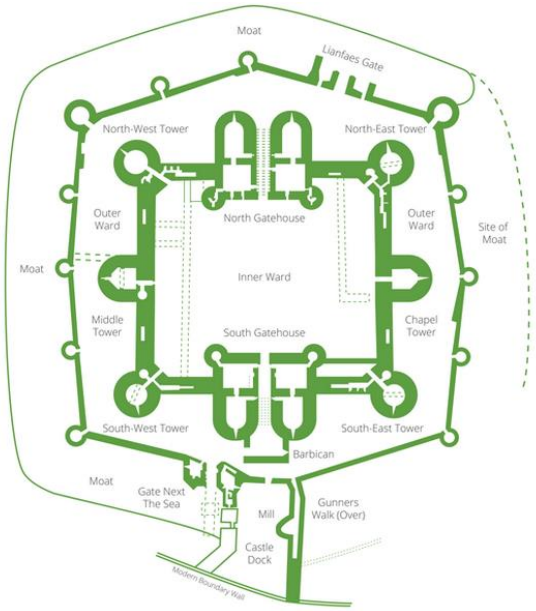
- Over 20yrs in the field
- Principal Cyber Engineer with Forcepoint
- Incident Response
- Worked in the past for the various military and government agencies
- Specialize in protection of critical infrastructure and DFIR

Protecting Your Network

- Traditional defensive posture
  - Maintain a strong perimeter
  - Implement layered security controls
  - Block known attacks and malicious IP addresses
  - Policies to discourage misuse or insider threat
  - Endpoint security products

Predictability Makes Us Vulnerable

- We know from the kill chain
  - Most breaches involve malware, phishing - **Human-based attack**
  - We still in many cases assume breach is going to come through the front door
  - Still focused on **“castle defense”**
  - We throw a lot of tech at the problem (shelf-ware)



Beaumaris Castle, 1295.
Source: Sucuri Blog

Target Breach

- Started on November 27, 2013
- Lasted 19 days
- 11GB of data stolen
- PCI-DSS compliant (Sept. 2013)
- 24/7 Dedicated security team (US/India)
- Tech from FireEye ($1.5M), Symantec, etc.

What Went Wrong?

- FireEye worked as designed
- Numerous “malware.binary” alarms
- Verizon found numerous methods to make it directly to cash registers
- Companies discover breaches through their own monitoring in only **31 percent of cases**.

“ Based on their interpretation and evaluation of that activity, the team determined that it did not warrant immediate follow-up.”

Molly Snyder, Target Spokesperson

“Dwell” Time

- Time a threat actor lingers until they are detected
- **200 day** average dwell time
- Home Depot = **5 months**

AMERICAN BANKER

Tuesday, March 3, 2015

Today's Paper | Magazine | Video | Web Seminars | White Papers — Women in Banking | FinTech Forward

DEALMAKING & STRATEGY | COMMUNITY BANKING | NATIONAL/ REGIONAL | LAW & REGULATION | CONSUMER FINANCE | BANK TECHNOLOGY | BANKTHINK

= Subscriber content; log in or subscribe now to access all American

Target Hackers Lurked for Months Before Pouncing at Holidays

by PENNY CROSMAN
FEB 11, 2014 12:18pm ET

It looks increasingly likely that the hackers responsible for the massive data breach at Target were lurking inside the retailer's network for months before they started swiping customers' credit card data, according to security expert and blogger Brian Krebs.

Bloomberg News

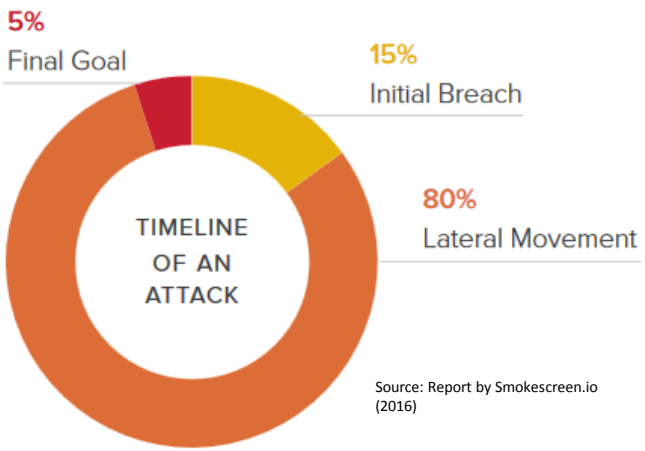
RELATED

Target Breach Cost to Banks: $172 Million — and Counting

Source: INFOSEC Institute – The Seven Steps of a Successful Cyber Attack

Detecting Lateral Movements

- Initial breach **normally doesn't yield value** to attackers
- Important part of the APT - hands on keyboard
- 80% of the attack is spent during lateral movement
- Your biggest win
- Attacker is moving blindly
- Easier to catch



Source: Report by Smokescreen.io (2016)

Typical Lateral Movements

- Goal is to stay under the radar
- Attackers use “legitimate” sysadmin tools
- Typical methods
  - Pass-the-hash (theft of NTLM hashes)
  - SMB scanning (i.e. file shares)
  - PowerShell scripts
  - Psexec
  - Windows Management Instrumentation (WMI)
  - RDP and other remote access (i.e. VNC)
  - Password brute-force

Traditional Honeypots

- What is a honeypot?
- “Decoy hosts” that are inherently insecure
- Designed to be attacked
- Imitate the activities of production systems that host a variety of services
- Gather information regarding an intruder into your systems

Traditional Honeypots

- Learn how intruders probe and attempt to gain access to your systems
- Gain insight into attack methodologies
- Learn how to better protect your real production systems
- Gather forensic information required to aid in the apprehension or prosecution of intruders

Interaction Level

Low:

- Enough interaction to attackers to allow the honeypot to detect attacks
- Mimics real services, limited logging, etc.

High:

- Full interaction with attackers to collect detailed information regarding the attack
- Real OS, real services, detailed logging, etc.

Honeypots

- Symantec Decoy Server
- Honeynets
- Nepenthes
- Honeyd
- KFSensor
- Cowrie
- Kippo
- Dionaea
- Conpot

IDS vs. Honeypot

- IDS reviews all traffic, events, etc. and based on predetermined signatures, policies - **Anomaly detectors**
- Makes a determination on whether events are indeed threats
- This leads to a lot of false positives and distrust by security operators

IDS vs. Honeypot

- Honeypots work on the premise that any triggered events are most likely not normal and should be investigated
- Reduces the amount of false positives
- Ensures operators are spending their time wisely

IDS vs. Honeypot

- SSH honeypot – given this isn't a "real" server, no one should be attempting to log SSH into it
- If a key is used where it normally shouldn't be, then we know that stolen keys are out there

What is a Canary?

"Order, configure and deploy your Canaries throughout your network. Make one a Windows file server, another a router, throw in a few Linux webservers while you're at it. Each one hosts realistic services and look and acts like its namesake. Then you wait. Your Canaries run in the background, waiting for intruders."

Source: OpenCanary Website

Architecture

- Canary is a commercial product built for internal use
- There is a free, open-source version called **OpenCanary**
- Mixed interaction honeypot
- Linux daemon that runs canary services, which trigger alerts following interaction

Source:
OpenCanary Website

Architecture

- Written in Python
- Developed using “Twisted” - event-based framework for Internet applications
- Uses other components/libraries (i.e. Redis)
- Free version lacks enterprise management tools

Modules

- FTP
- HTTP
- HTTP-Proxy
- MSSQL
- MySQL
- NTP
- RDP
- Samba
- SIP
- SNMP
- SSH
- Telnet
- TFTP
- VNC

```
root@kali:~# nmap -sS -A 192.168.0.119

Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2017-02-18 02:10 EST
PORT     STATE SERVICE     VERSION
445/tcp  open  netbios-ssn Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
1433/tcp open  ms-sql-s    Microsoft SQL Server 2014 12.00.4100.00; SP1
Host script results:
|_clock-skew: mean: 2d14h08m11s, deviation: 0s, median: 2d14h08m11s
| ms-sql-info:
|   192.168.0.119:1433:
|     Version:
|       name: Microsoft SQL Server 2014 SP1
|       number: 12.00.4100.00
|       Product: Microsoft SQL Server 2014
|       Post-SP patches applied: false
|       Service pack level: SP1
|_    TCP port: 1433
|_nbstat: NetBIOS name: SRV01, NetBIOS user: <unknown>, NetBIOS MAC: <unknown>
(unknown)
```

```
{
   "device.node_id": "foobar.com",
   "ftp.banner": "FTP server ready",
   "ftp.enabled": true,
   "ftp.port":21,
   "http.banner": "Apache/2.2.22 (Ubuntu)",
   "http.enabled": true,
   "http.port": 80,
   "http.skin": "nasLogin",
   "http.skin.list": [
      {
         "desc": "Plain HTML Login",
         "name": "basicLogin"
      },
      {
         "desc": "Synology NAS Login",
         "name": "nasLogin"
      }
],
```

```
"smb.filelist": [
      {
         "name": “CreditCard-Summary.pdf",
         "type": "PDF"
      },
      {
         "name": "passwords.docx",
         "type": "DOCX"
      }
   ],
```

HTTP Proxy Module

- Attacker needs Internet access to exfiltrate data so he searches for an open proxy
- Mimics either an MS-ISA or Squid proxy

```
"squid" : {
    # p/Squid http proxy/ v/$1/ cpe:/a:squid-cache:squid:$1/
    "banner": 'Squid proxy-caching web server',
    "headers": [
        ("Server", "squid/3.3.8"),
        ("Mime-Version", "1.0"),
        ("Vary", "Accept-Language"),
        ("Via", "1.1 localhost (squid/3.3.8)"),
        ("X-Cache", "MISS from localhost"),
        ("X-Cache-Lookup", "NONE from localhost"), # actually hostname:port
        ("X-Squid-Error", "ERR_CACHE_ACCESS_DENIED 0")
    ],
    "status_reason": "Proxy Authentication Required"
}
```

Triggers

- Access to a TCP service – Login attempt
  - HTTP, SSH, FTP, telnet, VNC, MySQL, MSSQL, RDP
- NTP
  - Issuing the “monlist” command (list of hosts that have connected to it)
- SIP
  - Any SIP type request
- Samba/SMB
  - Issuing a file read

Data

- Will vary depending on the module
  - Source IP / port
  - Destination IP / port
  - Local time / date
  - Node_id (if in a correlated environment)
  - Useragent (browser)
  - Remote client information
  - URL/Path
  - Credentials (most important)

Scenarios

- Let's look at three scenarios
  - **Cisco Device Discovery + Telnet Auth** – very typical for intruders to identify Cisco devices + auth attempt
  - **SSH brute force** – intruders will try to gain access to Unix hosts they find using weak passwords
  - **Access to Data** – intruders will look to steal sensitive data (i.e. PII, credit cards, IP, etc.)

Cisco Device Discovery

```
root@kali:~# nmap -sS 192.168.2.23
Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2017-02-18
06:05 EST
Nmap scan report for 192.168.2.23
Host is up (0.00017s latency).
Not shown: 992 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
23/tcp   open  telnet
MAC Address: 00:0C:29:12:02:C8
Nmap done: 1 IP address (1 host up) scanned in 0.14 seconds
```

Cisco Device Auth

```
root@kali:/home/pmorin# telnet 192.168.2.23
Trying 192.168.2.23...
Connected to 192.168.2.23.
Escape character is '^]'.

User Access Verification

Username: root
Password:
Authentication failed
Username: peter
Password:
Authentication failed
Username: larry
Password:
```

Cisco Device Auth

{"dst_host": "192.168.2.23", "dst_port": 23, "honeycred": false, "local_time": "2017-02-22 04:07:31.456133", "logdata": {"PASSWORD": "**password**", "USERNAME": "**root**"}, "logtype": 6001, "node_id": "opencanary-1", "src_host": "192.168.2.22", "src_port": 46992}

{"dst_host": "192.168.2.23", "dst_port": 23, "honeycred": false, "local_time": "2017-02-22 04:07:53.410849", "logdata": {"PASSWORD": "**woot**", "USERNAME": "**peter**"}, "logtype": 6001, "node_id": "opencanary-1", "src_host": "192.168.2.22", "src_port": 46992}

{"dst_host": "192.168.2.23", "dst_port": 23, "honeycred": false, "local_time": "2017-02-22 04:08:17.112158", "logdata": {"PASSWORD": "**hacked_password**", "USERNAME": "**larry**"}, "logtype": 6001, "node_id": "opencanary-1", "src_host": "192.168.2.22", "src_port": 46992}

SSH Brute-force

```
root@kali:/home/pmorin# medusa -u root -P /root/500-worst-passwords.txt -h 192.168.2.23
-M ssh
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [ssh] Host: 192.168.2.23 (1 of 1, 0 complete) User: root (1 of 1, 0
complete) Password: 123456 (1 of 499 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.2.23 (1 of 1, 0 complete) User: root (1 of 1, 0
complete) Password: password (2 of 499 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.2.23 (1 of 1, 0 complete) User: root (1 of 1, 0
complete) Password: 12345678 (3 of 499 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.2.23 (1 of 1, 0 complete) User: root (1 of 1, 0
complete) Password: 1234 (4 of 499 complete)
```

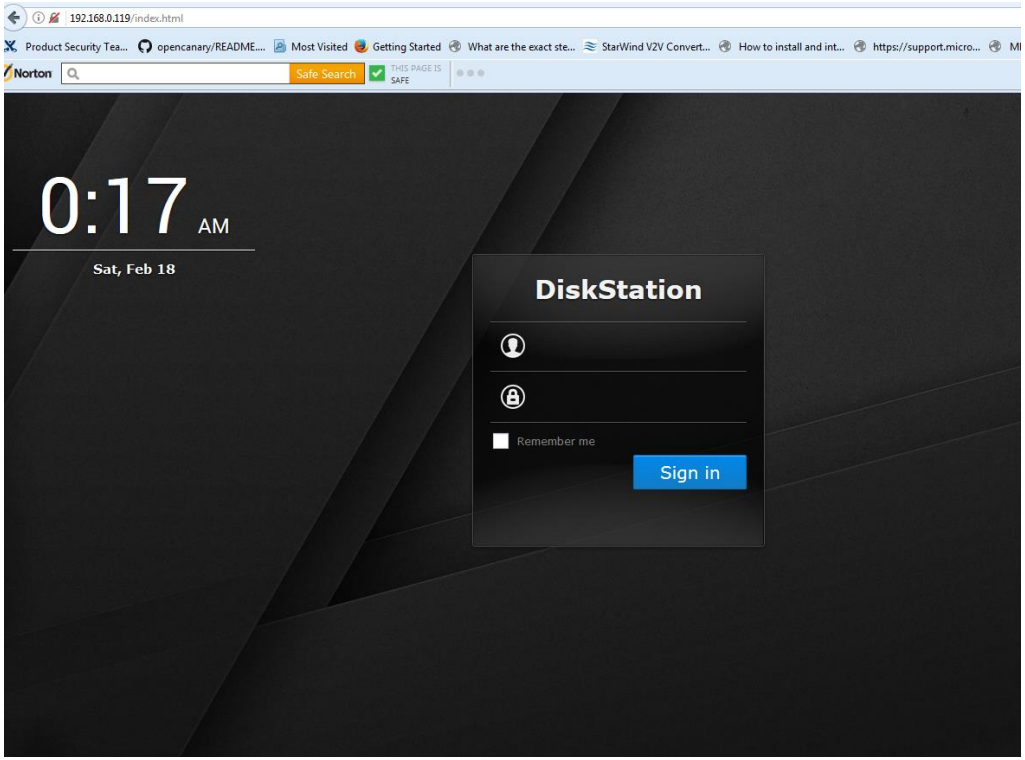
SSH Brute-force

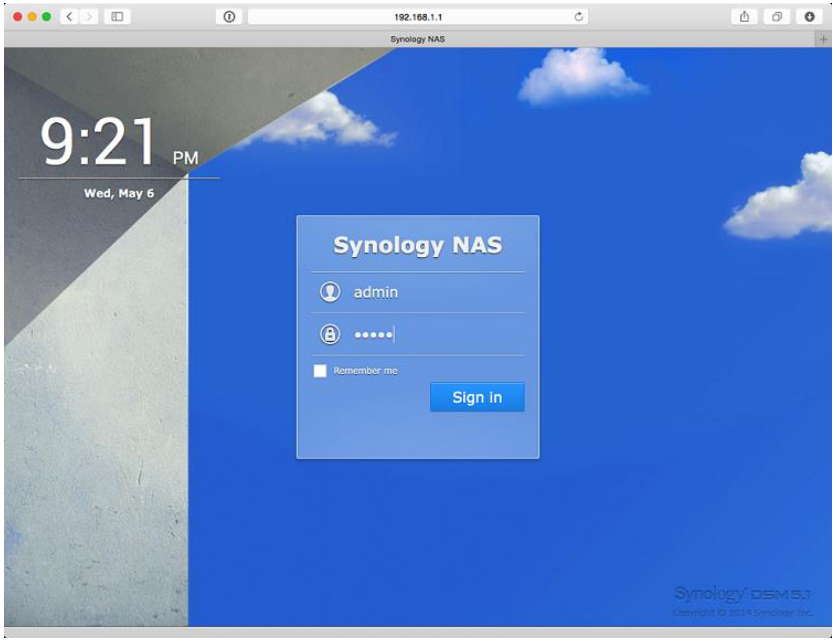
{"dst_host": "192.168.2.23", "dst_port": 22, "local_time": "2017-02-22 04:22:56.433314", "logdata": {"LOCALVERSION": "SSH-2.0-OpenSSH_5.1p1 Debian-4", "PASSWORD": "**123456**", "REMOTEVERSION": "SSH-2.0-MEDUSA_1.0", "USERNAME": "**root**"}, "logtype": 4002, "node_id": "opencanary-1", "src_host": "192.168.2.22", "src_port": 49040}

{"dst_host": "192.168.2.23", "dst_port": 22, "local_time": "2017-02-22 04:23:07.479054", "logdata": {"LOCALVERSION": "SSH-2.0-OpenSSH_5.1p1 Debian-4", "PASSWORD": "**password**", "REMOTEVERSION": "SSH-2.0-MEDUSA_1.0", "USERNAME": "**root**"}, "logtype": 4002, "node_id": "opencanary-1", "src_host": "192.168.2.22", "src_port": 49040}

{"dst_host": "192.168.2.23", "dst_port": 22, "local_time": "2017-02-22 04:23:18.522896", "logdata": {"LOCALVERSION": "SSH-2.0-OpenSSH_5.1p1 Debian-4", "PASSWORD": "**12345678**", "REMOTEVERSION": "SSH-2.0-MEDUSA_1.0", "USERNAME": "**root**"}, "logtype": 4002, "node_id": "opencanary-1", "src_host": "192.168.2.22", "src_port": 49040}

Access to Data





Access to Data

{"dst_host": "192.168.2.23", "dst_port": 80, "local_time": "2017-02-22 04:18:40.405712", "logdata": {"HOSTNAME": "192.168.2.23", "PASSWORD": "**diskstation**", "PATH": "/index.html", "SKIN": "nasLogin", "USERAGENT": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0", "USERNAME": "**root**"}, "logtype": 3001, "node_id": "opencanary-1", "src_host": "192.168.2.19", "src_port": 58455}

{"dst_host": "192.168.2.23", "dst_port": 80, "local_time": "2017-02-22 04:19:03.331599", "logdata": {"HOSTNAME": "192.168.2.23", "PASSWORD": "**password123!**", "PATH": "/index.html", "SKIN": "nasLogin", "USERAGENT": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0", "USERNAME": "**admin**"}, "logtype": 3001, "node_id": "opencanary-1", "src_host": "192.168.2.19", "src_port": 58455}

Valuable Forensic Data

- Where did the attacker traverse?
- How did they navigate to a particular network segment or host?
- What was the end target?
- Stolen SSH certificates
- Usernames/passwords that are used to attempt access
- Weaknesses are in your environment?

Alarming

- Send an e-mail using Mandrill
- Send an SMS using Twilio
- Output JSON to a TCP connection
- Write it to syslog
- Use other Python-based logging options
- Syslog to SIEM

Correlator

- Used to combine data from multiple OpenCanary sensors
- For example - individual brute-force login attempts - single alert via email or SMS
- Think of how a SIEM correlates events…

Deployment

- Don’t give too much away
- Build it like it was a real network
- A host shouldn’t be running everything
  - Cisco Router = Cisco telnet
  - Windows = MS SQL Server
  - Linux = Samba
- Placement is important (DB servers not in a DMZ)

Deployment

- Make this attractive to the attacker
  - DNS/host entries
  - Active Directory entries
  - Attractive hostname + following convention (i.e. DBSQL-EU01)
- Passive discovery techniques (over the wire)
  - NetBIOS name announcements
  - Simple Service Discovery Protocol (SSDP)
  - Cisco Discovery Protocol (CDP)

Deployment

- Vagrant
- Puppet
- Docker images
- Virtual machines
- Commercial product
- Raspberry PI deployment

Conclusion and Questions

- Think about embedding data within your environment that you want stolen for tracking (honey-tokens)
- Deployment design is key
- Think like the attacker
- Test the environment - adapt your processes
- Thanks to the folks at Thinkst (Marco Slaviero) for content for the presentation

Questions? Comments?

Peter Morin

petermorin123@gmail.com

Twitter: @petermorin123

http://www.petermorin.com