Detecting Infected Hosts and Domains

Tim Lim Dan Brown Ben Pusey

https://github.com/moneydance/591project

Abstract

Advanced persistent threats have become a major concern for IT professional around the world. Their stealthy distributed nature makes them difficult to identify and remove. However because the connections between backdoors and command and control centers leave tell tale traces in DNS data, certain graph theory techniques can be used to identify malicious domains, and infected hosts.

1 Introduction

Cyber security is an ever-changing field. As malware becomes more advanced so do methods of detection. Recently a sophisticated attack called an Advanced Persistent Threat (APT) has emerged. This stealthy form of malware attempts to blend in to the normal operations of the target organization. Fortunately due to the communication and infection vectors employed by APTs, tell tale patterns are left in an organization's DNS logs. In our implementation we leverage these communication patterns to detect suspicious activity that potentially indicates an APT infection.

2 APTs

2.1 Infection

APT's enter an organizations network through various means. USB's carrying malicious code, emails, and compromised websites are the most common vectors.

2.2 Covertly Spread

After infection the APT attempts to move through the network by infecting additional hosts. It does this covertly taking advantage of unpatched vulnerabilities and hijacked credentials.

2.3 Exfiltrate Data

After collecting the correct credentials and moving to the target hosts. The APT will begin to silently pass sensitive data out of the organization's network.

2.4 Call Home

In order to spread through the network and steal information the backdoor established by the APT must communicate and take commands from a hacker outside of the organization's network. Because organizations block inbound traffic, the communication must be initialized within the organization. To do this http/https connections are made from backdoors to command and control servers (C&C). These malicious domains are contacted at certain automated time intervals, asking for additional instructions from the server. Because these command and control domains are only contacted by infected machines their traffic is low. By finding rarely frequented domains and looking for patterns in the time intervals between a host contacting them we can label potentially malicious domains.

3 Methodology

Algorithm

1. Parse the logs into a graph. We build a bipartite graph where nodes consist of two sets, domains and hosts, edges represent a connection between a domain and host.
2. Run degree centrality on the graph to find domains with a low number of connections.
3. From these rare domains look for suspicious behavior. If the domain seems to be interacting with hosts in a suspicious manner mark it as a potential C&C domain.
4. Use these potential C&C domains and hosts connecting to these domains as a seed for our belief propagation algorithm.

3.1 Rare Domains

Malicious domains are contacted by a small subset of infected hosts. Because of this the degree of these nodes tends to be low. We can take advantage of this by using a degree centrality algorithm to discover rarely contacted domains. With this list of rarely contacted domains we can then look at the properties of the edges between the domains and the hosts that contacted them. We use two properties to define suspicious edges. The first is the time intervals of an individual host contacting the domain. If the standard deviation of the time intervals is low this indicates scheduled behavior (i.e a host was contacting the server for instructions on a regular basis). The second property is the number of CNAMES used by the DNS server. CNAMES map one domain name to another. A malicious domain could use multiple CNAMES to obfuscate the intended destination domain and make the original destination appear more legitimate. Using these two properties we are able to distinguish between legitimate low traffic websites and potential C&C domains.

3.2 Belief Propagation

We can use a belief propagation algorithm on a seed of malicious hosts and domains to iteratively grow our subgraph of infected hosts and know C&C domains. We do this by first looking at all the rare domains contacted by our infected seed hosts. If these domains exhibit malicious C&C like behavior we add them to a list of new malicious domains. If we have not detected any new malicious domains we begin belief propagation. We assign rare domains a score based on their properties being similar to known C&C domains. If this score is above a certain threshold we add the domain to our list of new malicious domains. After these two steps we update our known malicious domains with the newly found ones. We then update our malicious hosts with the hosts that contacted these new malicious domains, and update the rare domains we were looking at with the rare domains contacted by these new malicious hosts. We continue to do this until a certain number of C&C domains and malicious hosts are detected.

4 Experiments

Due to the sheer size of the data itself, running, let alone reading, all of it would require access to computers or large amounts of time that we simply did not possess. For this reason, it is a more suitable approach to look at a smaller amount of time, for example - a day's worth of data, instead of all the data that comprises several months of network information. Even just one day of data could contain a large amount of infected domains and hosts, and would pose quite a challenge in order to find those that may or may not be infected.

4.1 A day's worth of data

Many malicious hubs, malicious hosts, infected domains, and other infected hosts can appear within the time-frame of a day. Domains that were completely fine one day, can become infected on the next. So it is not unreasonable to test and do experiments involving just one day's worth of DNS log data. And in fact, looking at one day might be more beneficial than looking at an entire month, especially if later experiments also focus on just a day of data but later on in that month.
This could potentially form a basis for creating a more in-depth, focused, and localized area of interest within the possible infected sub-network(s) of the entire network, which could be more useful than a broad analysis involving an entire month of data.

5 Results

We run our code with the following settings, and output the sets of domains and hosts we find, as well as edge information between them, to a log file:

```python
infile = '2013-03-17'
num_edges = 500000
edgelist = parseToGraph.parse(infile, num_edges=num_edges)
G = construct_graph(edgelist)
hosts, doms = belief_propagation(G, set(), set(), threshold=0.7)
```

```
74.92.39.47
74.92.32.18
74.92.74.110
74.92.47.73
74.92.67.20
74.92.174.204
74.92.74.157
74.92.38.152
74.92.210.24
74.92.169.178
74.92.14.160
74.92.111.62
74.92.62.205
74.92.241.121
74.92.36.107
74.92.56.80
74.92.39.83
74.92.69.169
74.92.10.60
74.92.169.56
74.92.42.46
74.92.155.178
184.202.111.41
74.92.96.151
74.92.148.15
184.202.20.220
74.92.163.47
74.92.171.191
74.92.39.79
74.92.30.112
74.92.220.19
74.92.30.116
74.92.140.160
74.92.136.59
74.92.39.53
92.160.212.105
74.92.23.86
74.92.185.4
74.92.4.27
74.92.107.121
74.92.245.101
252.90.80.26
74.92.180.150
74.92.8.144
74.92.243.44
74.92.36.115
74.92.74.11
74.92.46.106
```

```
74.92.118.27
74.92.94.183
74.92.123.9
74.92.138.187
74.92.111.222
74.92.176.102
58.229.128.1
74.92.77.104
74.92.169.10
74.92.14.27
74.92.208.220
74.92.231.233
74.92.172.7
74.92.100.219
74.92.226.78
74.92.169.110
74.92.4.74
74.92.215.80
74.92.80.56
74.92.179.46
74.92.147.238
74.92.114.49
74.92.208.178
74.92.50.119
74.92.175.32
58.208.125.7
58.208.125.6
74.92.224.26
74.92.12.8
58.229.45.32
74.92.190.162
74.92.125.38
74.92.240.231
74.92.49.12
74.92.54.53
74.92.26.44
74.92.80.130
74.92.100.40
74.92.74.94
74.92.4.214
74.92.185.33
74.92.50.52
74.92.182.142
184.202.159.108
74.92.43.108
74.92.157.35
74.92.38.3
74.92.139.55
74.92.94.190
184.202.84.131
74.92.77.153
74.92.50.31
74.92.77.115
74.92.132.75
184.202.138.101
74.92.248.83
74.92.83.155
74.92.250.98
74.92.240.78
```

```
74.92.255.26
74.92.151.144
74.92.12.52
74.92.89.91
74.92.65.85
74.92.81.93
74.92.25.140
74.92.81.90
74.92.64.45
184.202.152.7
74.92.38.238
74.92.79.62
74.92.195.204
74.92.150.213
74.92.125.134
74.92.156.31
184.202.58.166
74.92.213.113

ump.thumb.dimly.wad
fulfil.johannes.wad
hastening.nullify.wad
suites.dusted.wad
cot.ledger.wad
rattiest.add.wad
lam.preponderances.wad
peddler.vet.wad
requisition.vanishing.wad
fa.fop.plot.hated.wad
gluey.jeans.tad.wad
shrimp.ab-z7g6r.noe
fa.ad-.plot.hated.wad
u.refurnished.wad
cot.fireproof.wad
oberon.aacire9v9zf.wad
pestilence.jocasta.noe
ob.enhanced.wad
step.hated.wad
braved.racier.wad
rev.guileless.wad
pit.clashed.rimbaud.wad
blantyre.superstitions.wad
bacteriologists.sapped.console.val
i.refurnished.wad
pours.comb.co.rd
dick.ably.ox.wad
ump.frill.dimly.wad
pert.la.agt.slyly.wad
sn.hated.wad
cot.preponderances.wad
na.blantyre.superstitions.wad
cot.ki.wad
ming.foam.inching.hated.wad
blantyre.rev.console.noe
occlusion.rimbaud.wad
rev.faint.wad
ripple.nails.wad
stoppering.conversationalists.wad
aaaa8y5807h1ayfufc0u7.tamra.b.tridents.noe
```

```
cot.jeans.tad.wad
rho.sedation.relentless.wad
kempis.jeans.tad.wad
pm.ohio.wad
cot.sledge.wad
fa.rue.plot.hated.wad
gent.ti.ty.friend.noe
did.toothy.co.rd
shadowiest.tad.wad
```

An example printout of the edge data encoded between a host and domain in this list is:

```
For nodes 74.92.32.18, braved.racier.wad:
date : 2013-03-17 00:05:38.806021
data:
? braved.racier.wad A
! braved.racier.wad CNAME collective.racier.wad
! collective.racier.wad CNAME collective.racier.wad.eco.racier.wad.friend.noe
! collective.racier.wad.eco.racier.wad.friend.noe CNAME marathon.racier.wad.wryness.noe
! marathon.racier.wad.wryness.noe CNAME moody.g.eyeballing.noe
! moody.g.eyeballing.noe A 59.195.249.218

date : 2013-03-17 00:10:38.720933
data:
? braved.racier.wad A
! braved.racier.wad CNAME collective.racier.wad
! collective.racier.wad CNAME collective.racier.wad.eco.racier.wad.friend.noe
! collective.racier.wad.eco.racier.wad.friend.noe CNAME marathon.racier.wad.wryness.noe
! marathon.racier.wad.wryness.noe CNAME moody.g.eyeballing.noe
! moody.g.eyeballing.noe A 59.195.249.218

date : 2013-03-17 00:15:38.834693
data:
? braved.racier.wad A
! braved.racier.wad CNAME collective.racier.wad
! collective.racier.wad CNAME collective.racier.wad.eco.racier.wad.friend.noe
! collective.racier.wad.eco.racier.wad.friend.noe CNAME marathon.racier.wad.wryness.noe
! marathon.racier.wad.wryness.noe CNAME moody.g.eyeballing.noe
! moody.g.eyeballing.noe A 59.195.249.218

date : 2013-03-17 00:20:38.755377
data:
? braved.racier.wad A
! braved.racier.wad CNAME collective.racier.wad
! collective.racier.wad CNAME collective.racier.wad.eco.racier.wad.friend.noe
! collective.racier.wad.eco.racier.wad.friend.noe CNAME marathon.racier.wad.wryness.noe
! marathon.racier.wad.wryness.noe CNAME moody.g.eyeballing.noe
! moody.g.eyeballing.noe A 59.195.249.218

date : 2013-03-17 00:25:38.757728
data:
? braved.racier.wad A
! braved.racier.wad CNAME collective.racier.wad
! collective.racier.wad CNAME collective.racier.wad.eco.racier.wad.friend.noe
! collective.racier.wad.eco.racier.wad.friend.noe CNAME marathon.racier.wad.wryness.noe
! marathon.racier.wad.wryness.noe CNAME moody.g.eyeballing.noe
! moody.g.eyeballing.noe A 59.195.249.218

date : 2013-03-17 00:30:39.008724
```

```
data:
? braved.racier.wad A
! braved.racier.wad CNAME collective.racier.wad
! collective.racier.wad CNAME collective.racier.wad.eco.racier.wad.friend.noe
! collective.racier.wad.eco.racier.wad.friend.noe CNAME marathon.racier.wad.wryness.noe
! marathon.racier.wad.wryness.noe CNAME moody.g.eyeballing.noe
! moody.g.eyeballing.noe A 59.195.249.218
```

6 Conclusion

In conclusion our methodology seems to be correct as a proof of concept. However for real world applications our system must be made more robust by being able to handle months of data instead of just a day. To do this we would have to build a software stack involving databases to filter out the normal DNS traffic of a company and find rarely contacted domains. Our methods for finding C&C domains could also be improved by expanding the parameters we are using to define suspicious activity and using more than the standard deviation of communication intervals between hosts and domains to identify automated domains.

References

[1] K. M. Carter, N. Idika, and W. W. Streilein. Probabilistic threat propagation for network security. IEEE Transactions on Information Forensics and Security, 9, 2014.

[2] T.-F. Yen and M. K. Reiter. Traffic aggregation for malware detection. In Proc. Intl. Conf. Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA, 2008.

[3] Alina Oprea, Zhou Li, Ting-Fang Yen, Sang H. Chin and Sumayah Alrwais. Detection of Early-Stage Enterprise Infection by Mining Large-Scale Log Data