APT INFECTION DISCOVERY USING DNS DATA

THE BASICS

Advanced persistent threat (APT) attacks on networks often consist of several initial stages, including:

1. Infecting hosts with an initial vector.
2. Downloading additional malware.
3. Establishing command and control communications.

The initial malware is typically severely limited in capabilities. Its primary focus is to exploit a targeted vulnerability, establish a privileged process, and download the next stage of the attack. In attempting to contact an external site, the malware leaves a record of itself in network flow and DNS logs. Once we are aware of the malicious nature of an outside IP or domain, it is generally a simple matter to determine which internal hosts were initially infected by examining our logs.

Similarly, while internal, lateral spread of compromise might go undetected, direct control of the infection also requires a revealing external connection. Again, once we identify the external command and control domains/IPs, finding the primary control hosts becomes much simpler.

In both of these cases, we rely on indirect means for determining whether a given external host is being used for malicious purposes. Internal and border detections are far from perfect, and even when detection does happen it may be lost in an abundance of false positives. Information from outside sources is useful when you aren't an initial target for an attack, but is often too late to be preventative.

THE CHALLENGE

To develop techniques for detecting malicious external hosts given the DNS logs for a site, and to identify potentially infected hosts in the process.

THE DATA

You will be provided with multiple months of DNS logs. These are real logs from a large site, sanitized to conceal their origin and the actual names referenced. Within these logs are name resolution requests from several simulated attacks. There could also be real attacks in these logs, though there shouldn't be anything large scale. The simulated attacks include initial callbacks to download additional malware, and command and control callbacks.

A data sample is available at ftp://ftp.lanl.gov/public/pflarr/. The rest of the data will be released shortly after EC3.

The code that originally parsed the data is at https://github.com/pflarr/dns_parse. The documentation and '-h' output thoroughly describe the data format.

LA-UR-13-23109

EXAMPLE DATA

These are the first few records in the sample data file with notes on each.

```
2013-02-03 12:50:36.003438,81.110.222.195,28.71.78.52,X,u,r,AA
? ra.urn.gus.gob.halloweens.elfin.how.interlinks.pad TEXT
! ra.urn.gus.gob.halloweens.elfin.how.interlinks.pad TEXT redacted
```
This is the response to a TEXT record lookup for a rather long domain name. The response may have given us some clue what was going on here, but as we have no effective way to sanitize the data it was redacted. When this occurred and the parties involved may still be significant, however.
```
<SEP>
2013-02-03 12:50:36.004926,84.190.147.34,28.71.78.1,X,u,q,NA
? .128.223.139.158.in-addr.arpa PTR
```
A standard reverse lookup. The first response is next.
```
<SEP>
2013-02-03 12:50:36.006238,28.71.78.1,84.190.147.34,X,u,r,AA
? .128.223.139.158.in-addr.arpa PTR
$ .86.233.in-addr.arpa SOA redacted
```
The response is the "Source of Authority" record for that IP address. It's where the client will need to ask next for more information. SOA's responses are redacted.
```
<SEP>
2013-02-03 12:50:36.008409,93.61.226.10,93.61.99.4,X,u,q,NA
? patronizingly.pang.raw A
```
A standard A (IPv4) query. It should become plain as you look through these that 93.61.0.0/16 is our primary network, and 93.61.99.4 is one of our main DNS servers. There are secondary /16 networks too that will show up as well.
```
<SEP>
2013-02-03 12:50:36.008810,93.61.99.4,93.61.226.10,X,u,r,AA
? patronizingly.pang.raw A
! patronizingly.pang.raw A 224.193.217.11
```
A quick response, (.004 seconds) and no secondary queries, mean the internal DNS server already knew the answer to this query.
```
<SEP>
2013-02-03 12:50:36.010402,93.61.226.10,93.61.99.4,X,u,q,NA
? .172.3.216.187.in-addr.arpa PTR
```
Another reverse lookup…
```
<SEP>
2013-02-03 12:50:36.010905,93.61.99.4,93.61.226.10,X,u,r,AA
? .172.3.216.187.in-addr.arpa PTR
! .172.3.216.187.in-addr.arpa PTR patronizingly.pang.raw
```
…about the same host as our A record as above, apparently.
```
<SEP>
2013-02-03 12:50:36.018600,93.61.18.139,93.61.99.4,X,u,q,NA
? kc.lean.knitter.bombs.pad A
```
Another standard A record lookup.
```
<SEP>
```

```
2013-02-03 12:50:36.018803,93.61.99.4,93.61.18.139,X,u,r,NA
? kc.lean.knitter.bombs.pad A
! kc.lean.knitter.bombs.pad CNAME vests.lean.knitter.bombs.pad
! vests.lean.knitter.bombs.pad CNAME knitter.cup.embraces.oat
! knitter.cup.embraces.oat A 102.6.60.96
! knitter.cup.embraces.oat A 102.6.89.168
“kc.lean.knitter.bombs.pad” is apparently an aleas for
“vests.lean.knitter.bombs.pad”, and that is an alias for
“knitter.cup.embraces.oat”. “knitter.cip.embraces.oat” can be reached at two
IP addresses.
<SEP>
2013-02-03 12:50:36.019078,93.61.18.211,93.61.99.4,X,u,q,NA
? meuse.jabs.aaaaaaaaaaaaaaaaaaaaaaa.philip.ipod.rim.pang.raw SRV
The “aaaaaaaaaaaaaaaaaa…” name is result of trying to find a replacement word
of that length (23!) and failing. The long sequence of random characters is
the replacement.
```

INDICATORS

In general, there is nothing outwardly indicative about the domains being queried or the format of the queries themselves. While DNS is being used in an entirely typical manner the names being queried, the timing involved, and the associativity of the activity is not normal behavior. Timing is key, and depends primarily on the initiator of the DNS lookup. APT lookups are initiated by one of three triggers: a host user, the malware itself, and the malicious actor.

USER INITIATED

The primary user initiated lookups are the initial infections. Whether it’s inadvertently browsing to a malicious site or clicking a link in an email phish, the look up will occur in the midst of other DNS activity initiated by the user. In practice, when looking at an instance of a host looking up the domain, the host’s recent DNS history can tell us a great deal about why the name might have been looked up. If the user got to a domain via general browsing, the surrounding lookups can tell us where else they went and what sites might have redirected them to (or at least linked to) the malicious domain. A clicked link in an email might not be surrounded by other browsing data, but there will be other lookups in the general timeframe that indicate that the user is actively using the machine at the time.

Kj;lkj

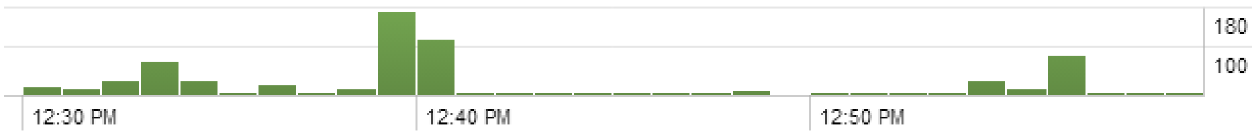

Figure 1 – It’s fairly easy to tell when a user is active.

MALWARE INITIATED

Malware initiated activity is generally at the mercy of an internal timer. Second stage downloads are likely to occur immediately after the initial infection, but further activity may not. Internal timers often dictate that the software wait until a certain time of day or a certain number of minutes before continuing. These timers are often used to evade detection by automated methods and resist reverse engineering attempts. They also serve to delay activity that may appear suspicious to the user until a time where they’re less likely to associate it with its actual cause.

What this means is that malware initiated DNS activity is just as likely to be seen at times when a host isn't being used as not. There won't be a relationship between user activity at these requests, or at least not one that is readily apparent.

MALICIOUS ACTOR INITIATED

Once an outsider has gained control of a host the attacker can proceed in several different directions, often at once. Additional tools may be downloaded to the host, data from the host may be gathered and sent externally, and other hosts may be attacked laterally. Like malware initiated activity, these events will occur without regard to whether the host is being used by its local user. Attackers do occasionally limit their activity to within local working hours, however.

Additional tool downloads often occur over existing communication channels, but that isn't always the case. These tools may also simply be common and publicly available; it's somewhat strange for a random host on the network to download a password extraction tool, but not in the sense that we're looking for here. Proprietary tools are likely to be more protected and selectively transferred, and as such they tend to be fetched from more directly controlled sites.

Data exfiltration (gathering and sending data externally) is typically sent to a separate external staging area over separate communication channels. What's interesting about this step is that it should have a profile similar to prior steps in the infection process. While it might not be useful to look for this in particular, methods may find it simply as a side effect.

Lateral movement (infecting other hosts within an already infected network) is typically achieved by mining a host's memory for credentials that are valid on other hosts on the network. Given those credentials, an attacker can hop to other hosts, grab credentials there, and so on. Eventually, the attacker hopes to gain access to more valuable hosts either in terms of their function or contained data. This hopping from host to host is apparent in DNS logs, unfortunately the internal connections themselves are normal almost by definition; such connections are often how the credentials, especially admin credentials, got on the original host in the first place. Associating hosts for detecting lateral movement is possible, and some success has been shown in doing so. Using this as an active detection system fails simply due to the rarity of actual occurrences.

EXAMPLE TIMELINE

1. A dozen employees receive a well-crafted phish email with an enticing link to a malicious site (terrible.au).
2. (Minutes to hours later) The recipients see the emails and some click the link. The loaded page detects their browser and OS version, and redirects them to a new host (zaphod.be) with a targeted payload.
3. (Moments later) On some systems the malicious payload is successful. Compromised browser downloads (also from zaphod.be) and executes the malware second stage from a secondary location.
4. (Twenty minutes later) The second stage begins beaconing home to a third host (gig.bacon.ca).
5. (Several hours later) The malicious actor sends the command over the beacon (command and control) channel to establish a tunnel to the beacon host over http.
6. (Over the next several hours) The malicious actor gathers documents from that host, and sends them out over https to an external staging area (gord.third.clam).
7. (Over the next few weeks) The malicious actor moves lateral to different hosts in the network, proxying through the initially infected host. Documents gathered and sent to the staging area directly, however.
8. (Indefinitely) Backdoors are established on additional hosts (not all are utilized), and data gathering continues.

Keep in mind that this is simply one of example of the numerous ways an incident could proceed. Different targets, attackers, and objectives may yield fairly different results.

CASES

While the overall goal of this challenge is to look the data as a whole (or as on incoming stream) and identify potentially malicious domains and compromised hosts, we've constructed a set of cases to ease you towards that goal. Initial cases include significant indicators to help you know where to look. As the cases progress, the goals become broader and outside information evaporates.

In amongst the normal DNS traffic are queries added artificially to simulate the queries an intruder would make in a real attack. Each of the following cases will include multiple instances; the exact details of each will be included with the data. Answers are provided for each instance. It is recommended that any algorithms be tuned for false positives against a subset of these instances, and then tested against the rest.

It's often asked how we would go about solving any of these cases manually. The answer is that we don't, or at least don't in a way that is relevant here. Case 1 is most like our normal procedure, in that we start from entirely separate indicators and then use already known DNS information about the case to try to identify other local hosts that might have been compromised. This information can come from internal indicators, or indicators from other, non-local sources.

CASE 1: FIND THE MALICIOUS DOMAINS.

For the first case you will receive a tip concerning a host that has been talking communicating with a malicious IP address and the approximate time the communication started. From this you will need to find the malicious domains used in the attack, only some of which will be connected with the malicious IP given.

CASE 2: FIND THE ATTACK GIVEN A SET OF KNOWN COMPROMISED HOSTS.

In this case you will be given a set of hosts that appear to have been compromised a rough timeframe of when this occurred. Your goal is to find the domains used in the initial infection, callbacks, and command and control channels.

CASE 3: FIND THE ATTACK GIVEN ONE SUSPICIOUS HOST.

As per the Case 2, but you will only be given one host that may be compromised as a starting point. An additional goal is to find all the other hosts compromised in the attack.

CASE 4: IDENTIFY POTENTIALLY MALICIOUS DOMAINS

Your goal is to identify domains that may be involved in an attack. You will not be given any indication of the hosts that may be compromised, but you will know when they occurred (within a given day). No more than 5 false positive malicious domain identifications for a given day.