Typosquatting Feed data for a DNS firewall

Posted on May 5, 2021

There is a tremendous number of domain names registered daily which resemble legitimate domains of brands or organizations, or whose names imply being related to a known service or product. Domains suggesting to be a “support” or “account-verification” or “support page” are also common for containing such strings. Initially, many of these are parked and some become used in malicious activities such as being sold at an inflated price to the legitimate owner, being used as botnet Command & Control servers, or in phishing campaigns to host fake pages to have the victim's sensitive data typed in and sent to the miscreants.

It seems trivial to remind users of the Word Wide Web that they should be very cautious when following a link they have received in a message. But it is not the case: as the Web is used by people from diverse cultures, education, and technical backgrounds, phishing can harvest its victims. To say nothing of botnets: a bot is a hidden piece of software and thus it uses DNS queries to find its C&C server without any notification to the user.

The Typosquatting Feed detects groups of similarly-named domains registered on the same day. The so-arising list of domains definitely needs attention as it contains a number of malicious or potentially malicious ones. To give an example, on 2021-02-22 the feed contained the following two triplets of domains:

paypal-securecenter[.]shop
paypal-securitycenter[.]org
paypal-securitycenter[.]shop
paypal-safetycenter[.]org
paypal-safetycenter[.]shop
paypal-safecenter[.]shop

And indeed, on 2021-04-14, IBM X-Force Exchange announced an early warning about a squatting campaign against PayPal in which these domains were involved. In spite of the possible false positives, domains listed in the Typosquatting Feed are better treated as suspicious and blocked unless there is a good reason to whitelist them. But there are close to 10 thousand domains there in a text file every day. How can one prevent the sites under these domains from being opened, or, more generally, prevent them from name resolution at least locally? The answer is as follows.

1. DNS Response Policy Zones and DNS firewalls

The DNS infrastructure's basic elements are the nameservers that build up the global distributed database of domain name resolution data. (See our DNS primer for details.) Consider, as an example, a simple caching name server serving a local network. It serves machines in the local network by getting DNS information from external name servers, caching them, and distributing them locally. A DNS Response Policy Zone (RPZ) enables the administrator of a name server to modify the response of the name server in real time.

So, for instance, if a domain badguys[.]evil is known to be a malicious one and thus is included in the RPZ, then instead of sending back its IP to the client it can, for instance, reply as if the domain didn't exist or resolve it to an IP where a local warning web page is displayed to the client. So,while the original information in the DNS is intact, it is represented appropriately according to thelocal policy. Such a setup is a kind of firewall: clients using this nameserver will get a responsethat the domains do not exist whenever they ask to resolve a blacklisted domain.

2. Implementation

Let us give a basic illustration of how to set up such a DNS firewall using the data of the Typosquatting Feed. We set up a simple caching nameserver on a Linux box (running Ubuntu 20.04.2 LTS; the process would be rather similar on any other Linux or BSD system), and introduce the RPZ using the data of the Typosquatting Feed. This can be readily used locally, but a more elaborate name server serving e.g. an organization's network can also be set up along these lines.

The process is simple, we need only a few steps.

2.1 Set up a caching name server

Install bind9:

```
sudo apt-get update
sudo apt-get install bind9
```

With the default settings, it will work as a caching name server. (For setting up a more elaborate config on a Debian-type system, you may consult e.g. this reference. But the default settings are perfectly suitable for our demonstration.)

You can verify that it works by making a query using your localhost as a server, e.g. to resolve whoisxmlapi.com:

```
dig @localhost whoisxmlapi.com
```

2.2 Configuring Response Policy Zone (RPZ)

Add the following lines to the end of the options Section in /etc/bind/named.conf.option:

```
//enable response policy zone.
response-policy {
zone "rpz.local";
};
```

(Note that normally there will be a }; after this, as this is the part of the OPTIONS{ section.)

Then in /etc/bind/named.conf.local we add the respective zone db, like this:

```
zone "rpz.local" {
  type master;
  file "/etc/bind/db.rpz.local";
};
```

For more elaborate configuration, including logging, etc. it is recommended to consult e.g. this tutorial. But the above minimal setup will actually work; it only remains to provide /etc/bind/db.rpz.local, using the Typosquatting Feed data.

2.3 Setting up local RPZ based on Typosquatting Feed data

So, our goal is to convert at least a part of the Typosquatting Feed's data to a local RPZ database ensuring that our name server will resolve all the listed TLDs and their subdomains as if they were non-existent.

We assume a monthly subscription to the Typosquatting Feed. This is sufficient as most domains that become malicious do it with a few months of delay after their registration. The Typosquatting Feed data are available via Web or ftp download. Here we assume that the data files are downloaded regularly and they're stored locally in the directory /data/typosquatting_feed.

To convert them into a local RPZ database, we use the following script, intended to be run as root:

```
#!/bin/bash

TYPOSQUATTING_FEED_DIR=/data/typosquatting_feed
WHITELIST=whitelist.txt
CURRENTLIST=current_list.txt
DBFILE=/etc/bind/db.rpz.local
NMONTHS=2

temp1=$(tempfile)
temp2=$(tempfile)

the_date=$(date +"%Y%m")01

for m in $(seq 0 $NMONTHS);do
   typosqdate=$(date -d ${the_date}-${m}month +"%Y-%m-%d")
   typosqfile=$TYPOSQUATTING_FEED_DIR/typosquatting.$typosqdate.monthly.full.basic.csv
   echo $typosqfile
   if [ -f $typosqfile ];then
      echo "processing for $typosqdate" >&2
```

```
        cat $typosqfile | cut -d, -f 5 >> $temp1
    fi
done
sort -u $temp1 > $temp2
sort -u $WHITELIST > $temp1
comm -13 $temp1 $temp2 > $CURRENTLIST
rm $temp1
rm $temp2

serial=$(TZ=UTC date +%Y%m%d%H)

cat > $DBFILE <<EOF
; RPZ zone for the local nameserver
\$TTL 14400
@       86400   IN      SOA     localhost. root.localhost. (
$serial         ; serial, todays date+utc time
86400           ; refresh, seconds
7200            ; retry, seconds
3600000         ; expire, seconds
86400           ; minimum, seconds
      )
@     IN        NS      localhost.

EOF

awk '{print $1"\t CNAME .\n*."$1"\tCNAME ."}' < $CURRENTLIST >> $DBFILE
named-checkzone rpz $DBFILE
retval=$?
if [ $retval -eq 0 ];then
    echo "New config stable, restarting name server" >&2
    systemctl restart bind9
else
    echo "Incorrect $DBFILE produced, not loading." >&2
```

```
fi
```

The script is configured at the beginning:

TYPOSQUATTING_FEED_DIR
holds the Typosquatting Feed's downloaded data files.

WHITELIST
specifies a file which has one domain in each line that will be resolved under any circumstances. This whitelisting ensures that the critical services will be available.

CURRENTLIST
is a text file that shows the list of currently blacklisted domains for reference.

DBFILE
is the local RPZ zone file that we create or overwrite.

NMONTHS
is the number of months to process in addition to the current one. The current setting of 2 will result in the processing of 3 months altogether. Obviously, the greater this number is, the more rigorous the server will be. However, the 3 months will result in a zone file of 50-100 megabytes, and BIND will keep it in memory, indexed. The present script was tested on a virtual host with 2 gigabytes of memory; the installed DNS server managed to serve a small private network this way, but for more months there is more RAM needed.

The rest of the script works as follows:

- It puts together the list of domains in the respective Typosquatting Feed files in a temporary file
- It removes the whitelisted domains from this list.
- It converts the list to the format of a zone file.
- It checks if the generated zone file is correct.

- It restarts BIND, the name server software.

All you need to do is to download the typosquatting files, add at least a few lines to the whitelist, and run the script on the name server that was prepared as described. This should then be repeated once a month, typically on the first Tuesday: add the new monthly Typosquatting Feed file, and run the script as root.

As a result, one gets the following result when querying localhost about one of the suspicious domains:

```
kmatyas@localnameserver:~/binduptdate$ dig @localhost paypal-securecenter.shop

; <<>> DiG 9.16.1-Ubuntu <<>> @localhost paypal-securecenter.shop
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 33039
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 3e7a469f062ac0b30100000006078211b29bc7cd0ffe6312b (good)
;; QUESTION SECTION:
;paypal-securecenter.shop. IN A

;; ADDITIONAL SECTION:
rpz.local. 1 IN SOA localhost. root.localhost. 2021041511 86400 7200 3600000 86400

;; Query time: 236 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Apr 15 11:18:51 UTC 2021
;; MSG SIZE  rcvd: 140
```

Notice the "ADDITIONAL" fields: they reveal that the reply is provided using our blacklist. Now if a phishing mail comes with a link to this domain, all clients using this nameserver will get the reply that this domain does not even exist.

3. Outlook

Our demonstration is rather basic, yet it can be readily used e.g. on a Linux workstation, laptop, or on a server serving a home or small-business network. Certainly, to do it properly, one has to set up a secondary name server and enable the transfer of the RPZ database to make the service robust. The way of doing it is described in this tutorial. The Typosquatting Feed has false positives, too, and this is partially handled by our whitelist. In a larger network, however, logging can help with identifying the domains which were actually queried, and whitelisting the benign ones. One may also consider modifying the solution so that a blacklisted domain or its subdomain resolves to a local warning page to inform the user and ask her/him to contact the administrator to whitelist the domain if it is benign. We have found in practice that such occasions are rare.

Using DNS firewalls is an efficient way for protecting a network against certain scams, notably phishing and botnets. As demonstrated here, the Typosquatting Feed serves data which contribute to the quality of such a service.