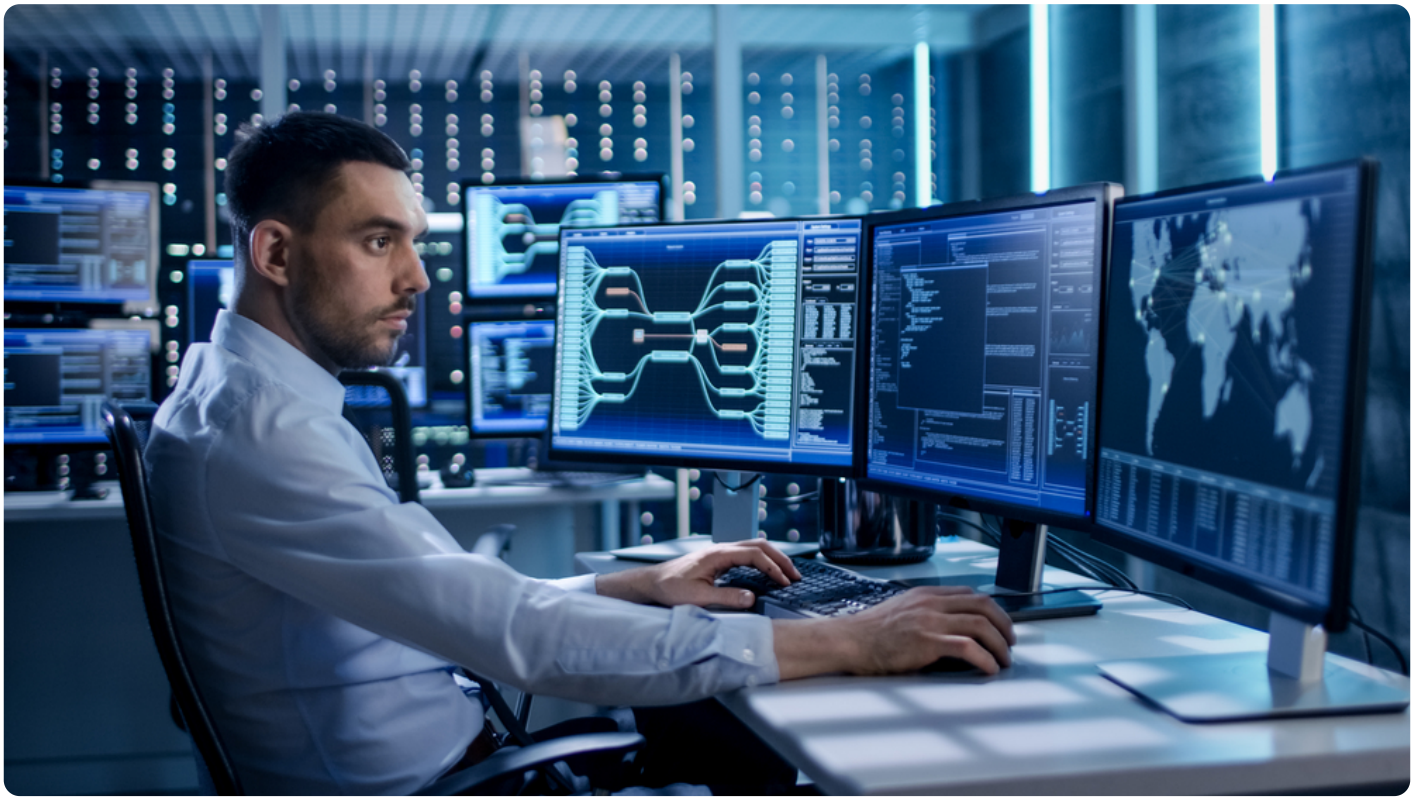
Essential tools for server operators in action

Posted on May 13, 2020



The Internet is a very dangerous place. A server with a public IP address could become the subject of an attack virtually at any time of its operation. Indeed, any service that is vulnerable to any extent is likely to be exploited at some point if left this way; no server operator can deny playing this cat-and-mouse game with hackers.

Not all hackers meet the stereotypes attributed to them, though. One does not have to be an ingenious IT specialist with very tricky ideas to try and exploit servers. Picking an exploit kit written by someone else, and letting it run on arbitrary IP addresses is essentially free, and it will surely harvest something: sooner or later it will run into a content management system on a website whose owner failed to apply some important security update, or web-based database management console left open to the public. All these could result in an administrator's access to the server, which may lead to dramatic consequences for the owner.

It is always instructive, for instance, to frequently take a good look at the access log of your web servers. Let us conduct a bit of an investigation to illustrate what is typically going on.

Below you can see lines from the access log of an nginx web server, running a small, private webpage on a standalone Linux-based server somewhere in Europe (this will be important later). It has no PhP support, and just serves the simplest and purest old-fashioned html pages. Hence, if someone tries to access e.g. anything in php, this must be an exploit.

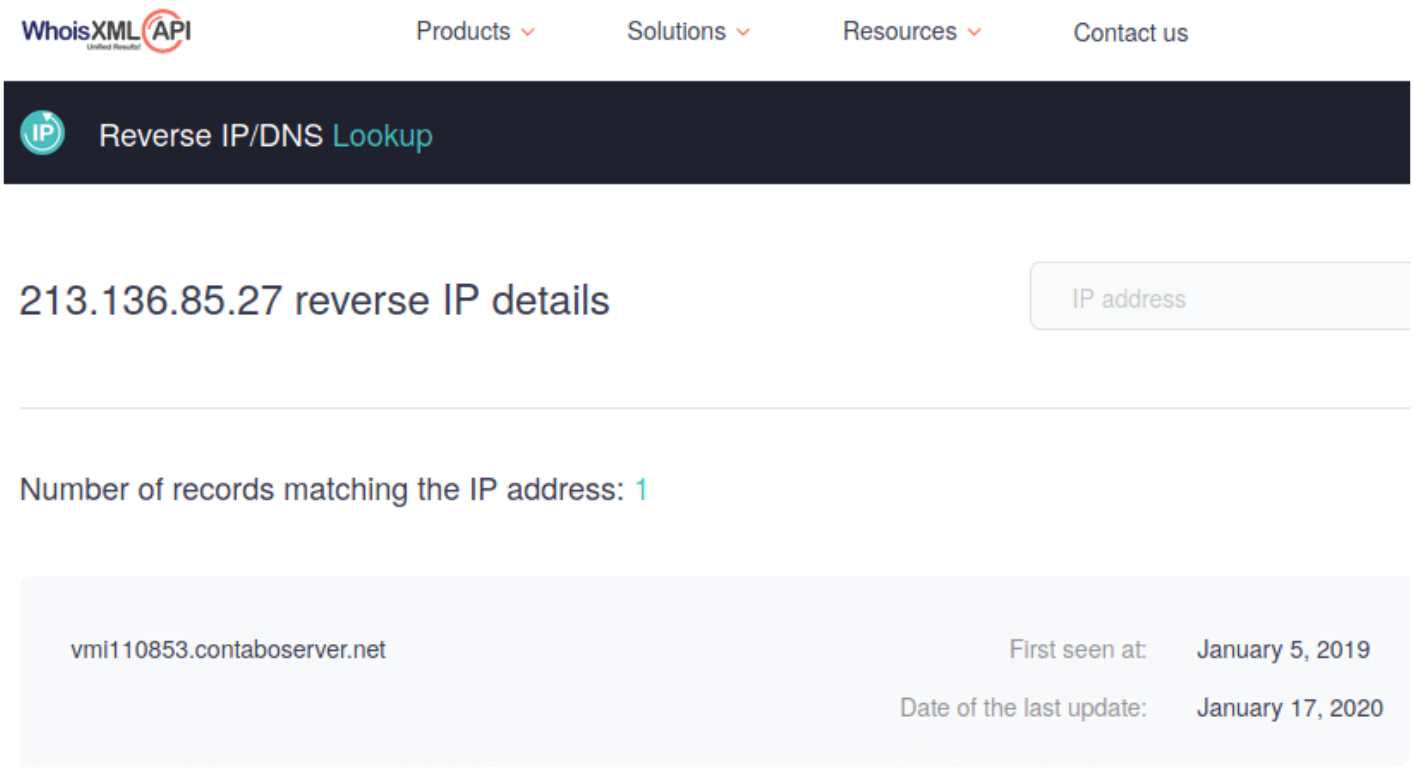
And indeed, on the day of writing this blog, we find the following lines with the string "php" in the access.log file:

```
58.210.85.22 - - [25/Feb/2020:04:30:17 +0000] "GET /phpMyAdmin/scripts/setup.php HTTP/1.1" 404
58.210.85.22 - - [25/Feb/2020:04:30:18 +0000] "GET /phpmyadmin/scripts/setup.php HTTP/1.1" 404
58.210.85.22 - - [25/Feb/2020:04:30:18 +0000] "GET /pma/scripts/setup.php HTTP/1.1" 404 143 "-" "Z
58.210.85.22 - - [25/Feb/2020:04:30:20 +0000] "GET /myadmin/scripts/setup.php HTTP/1.1" 404 143
58.210.85.22 - - [25/Feb/2020:04:30:22 +0000] "GET /MyAdmin/scripts/setup.php HTTP/1.1" 404 143
213.136.85.27 - - [25/Feb/2020:11:10:03 +0000] "GET /phpMyAdmin/scripts/setup.php HTTP/1.1" 404
213.136.85.27 - - [25/Feb/2020:11:10:03 +0000] "GET /phpmyadmin/scripts/setup.php HTTP/1.1" 404
213.136.85.27 - - [25/Feb/2020:11:10:03 +0000] "GET /pma/scripts/setup.php HTTP/1.1" 404 143 "-"
213.136.85.27 - - [25/Feb/2020:11:10:03 +0000] "GET /myadmin/scripts/setup.php HTTP/1.1" 404 143
213.136.85.27 - - [25/Feb/2020:11:10:03 +0000] "GET /MyAdmin/scripts/setup.php HTTP/1.1" 404 143
```

"Our friends" do not want to even hide that they are using ZmEu, a rather outdated vulnerability scanner which tries to launch an attack via the very popular phpMyAdmin program. This can in fact be also found out from the pages they try accessing - as said before, in the lack of any support for PhP on this server, without any success.

But who are they? Of course it is much better to ask questions like this from behind the shelter of a server which is rather secure because of its simplicity and limited services; we are just calmly looking at the unsuccessful attempts in our server's log files. But imagine that it is your server, running some business critical and important service, and that some attack like this has been successful… You would probably ask this question much more impatiently: who are they?

Well, the standard idea would be to check the domain name behind the two distinct IPs in the above list. Sure enough, we are in need of a reverse DNS lookup, which reports host names assigned to an IP address. With the Reverse DNS/IP lookup provided by WhoisXML API, in case of one of the addresses, it works out:



The same hostname can be found with the host Linux/Unix command or some other DNS utility, except that we do not get "First seen" or "Last update" data. These dates come from passive DNS, a very essential ingredient of cybersecurity which collects its data from actual DNS lookups, and therefore is not restricted to the actual data of the Domain Name System. Returning to the host name we have found, a WHOIS lookup with WhoisXML API's tool returns a fully masked registrant, but at least the registrar, a big company from Germany is revealed. So if we were to investigate the incident further, we would have the initial clue to find our attacker: a company interested in domain registration will help us as it is also detrimental for them to host an attacker.

How about the other IP address? Unfortunately no active or passive DNS lookup returns any result. What does this mean? The attack came from an IP address which has not been assigned a name in the Domain Name System, and was not detected in any actual communication. What can we do? Here is where the IP netblocks WHOIS data come into picture. The space of IP addresses is subdivided into a hierarchy of intervals, termed "netblocks", which are assigned to entities who own them and are responsible for them. So even if an IP address does not have a host name

assigned, we can still figure out the ownership of the netblock it belongs to.

This is doable with the IP Netblocks API. Investigating a single case, all we need to do is to go to its webpage and enter the IP address. The result will be a set of IP number ranges our suspect is an element of, and the list will start with the most interesting one, which is the narrowest of all:

WhoisXMLAPI

IP Netblocks API

API docs

Preview XML JSON

IP range(s) found: 6

Report price: 1 credit

Inetnum: 58.210.85.16 - 58.210.85.31

Inetnum First: 986862864

Inetnum Last: 986862879

Autonomous System

Autonomous System Number: 4134

Name: China Telecom

Type: NSP

Route: 58.208.0.0/12

Domain: http://en.chinatelecom.com.cn/

Netname: suzhou-SUZHOU-XIANJIAODA-CORP

Description

suzhou xianjiaotong univ fazhan co.,ltd

suzhou city

jiangsu province

So this attack came from a netblock managed from China this time. If it had been to succeed, here is the very clue to follow to get to the attacker: a telecom company of good reputation will surely collaborate with us and the authorities to find the miscreant who is actually deteriorating their own reputation, too.

So if we were in an unpleasant situation to investigate after a successful attack, assuming that we are still in hold of the log files, we could find our way to the miscreants - to the level where it is a well-prepared case for the police. But not only this: we have learned about our enemies. And for sure, they are enemies as the owner of the server did not have any agreement with anyone to test his infrastructure against these exploits, so these were real intrusion experiments with certainty. And the target was a very marginal little private web server in Europe. Even such a thing could be attractive for a cyber criminal: if exploited, it is a resource for further dirty activity… But, as demonstrated, there are countermeasures to apply.

On a critical server running an elaborate infrastructure, it is a good idea to do similar investigations in real time. For instance, if one could keep a honeypot web server and detect such an attack, the IP addresses could be fed back to the firewalls immediately. WhoisXML API's services are perfectly suitable for such solutions as all the illustrated lookups, and many more, are available as handy RESTFul APIs which can be integrated to scripts and other pieces of software. For protecting a bigger infrastructure, local databases of IP Netblocks data , DNS history data or Domain WHOIS data can be built and maintained using WhoisXML API's data download products to facilitate high-throughput queries. With the appropriate tools, your online services can be made as secure as your most innocent and naive user would imagine how the Internet should be.