Les Chroniques du Campagnol
Season 2 Episode 3
Presenting the Workstation Status Uploader

April 2, 2022

My next challenge for Season Two of the Campagnol Chronicles is to write an edge computing module to gather and share information about my workstation and its neighbourhood.

1 But the most important thing to do first

Is of course to find a moniker for the application. After a few errands, I settled on MERCATOR. Seems about right for something that will chart known territories in a consistent albeit slightly distorted manner.

2 User Needs

The user, in this case, is me. I need to reliably identify and connect to different workstations depending on the circumstances, and especially on the current conditions of the network I'm currently connected to.

2.1 Connect to Polarion

I need to connect to Polarion from my Windows workstation. However, when I'm working from home, I'm not always connected to the VPN. I do not connect systematically because, mainly, I don't need it that often and it sometimes interferes with the DNS resolution, making for example Outlook unable to connect to the mail server. When I need to connect to Polarion from home, one of these three possibilities exist:

- My workstation is physically connected to the DINA network because I'm working from the office. This case is so trivial that I will not discuss it further.
- The VPN is active on my workstation. In that case, well, it's simple, I can connect to Polarion normally as if I was at the office. After all, that's the whole point of a VPN.
- The VPN is not active on any workstation in my home network. In that case, well, it's simple also, I just cannot connect to Polarion, and I need to somehow activate the VPN.
- The VPN is not active on *my* workstation, but it is active on *another* workstation that happens to be acting as a proxy.

  Aha! You said ***three*** *possibilities, but you listed* ***four****!*

No, I truly did list **three** possibilities. Re-read the text carefully[1]. Of these, the last one is obviously the most challenging, but we need to provide solutions for all three, including the one where Polarion is not reachable at all.

You are going to write a proxy! How nice!

[1] I bet that you dismissed the third item as an *impossibility* rather than a possibility. Unfortunately that is not the correct answer. The one to disregard is the first one, because I explicitly said that I was working *from home*. Anyway, the first and second items are fundamentally the same thing, as using the VPN installed on my machine or the one in a router at the office are exactly the same thing, I am *logically* connected to the DINA network.

No, I'm not. The proxy already exists, it's just about leveraging the *nginx* front-end I use when I'm running a LabX backend server on my development laptop[2]. I've covered this topic in one of the very first installments of the Campagnol Chronicles[3]

An obvious solution is to use *two* shortcuts to access Polarion, one that uses `http://polarion.dina.local` as the URL, and a second one that uses `http:/sle-lp11`. Or even explicitly use the IP address. Case closed.

But I suppose that if you start a series of Campagnol Chronicles that are likely to span at least a few months, that obvious solution obviously does not work as smoothly as one can hope.

Indeed! One of the issues I face is that I cannot connect to Polarion using any name I fancy, I *must absolutely* use `polarion.dina.local` or `polarion` because the links inside a Polarion page explicitly use that name in the URL instead of using relative html references. Anyway, Polarion links can be found everywhere, in e-mails or Slack messages, in release notes, in knowledge base items, in the development Wiki. Everywhere, they can be everywhere, and I must be able to both use a link that was sent to me, or send a link to someone else.

So you need a custom DNS server. Case closed.

Yes and no. Yes I need customisable name resolution because I must resolve the name to either the "real" Polarion server (in the case I have VPN access), to the IP address of the PC I use as proxy (in the case where I don't have direct access but my proxy does), or even to a fake Polarion server (in the case where neither my workstation nor my proxy PC has access).

But no because I will not merely install and configure a third party DNS server; after all, *that* is merely a junior DevOps routine, not a programming challenge.

And also no because if I install a DNS server, I'm back at square one: now I have to change the DNS configuration depending on whether I'm on the network where that DNS server is available or not.

And also no because that is only the emerging part of the iceberg. The real issue is that my workstation can change from one network configuration to another several times, ie each time I start or stop a VPN connection on one of the devices on the network. So I need to use a *dynamic DNS* so that address resolution can be changed on the fly. **And** I still need to write an application that detects my current situation and updates the DNS configuration.

And finally no because I don't actually need to install a DNS server: there is already a custom DNS server natively built in Windows (and in Linux also, by the way). I just need to write the tool to configure it properly and automatically.

Nah. I've never heard of a built-in custom DNS server in Windows. There is of course one in Windows Server, but not in Windows 10.

Well, yes, there is one. It's there since *Windows 95*, and maybe even in *Windows for Workgroups 3.11* if you were bold enough to install *Trumpet Winsock*. You've heard of it but you don't think of it as a DNS server because its just the `hosts` file. A nice feature of a DNS server configured by a simple text file is that, precisely, you only have to write the proper contents in the configuration file and voilà, the DNS is configured. No need to implement fancy protocols that can fail in mysterious ways or become obsolete overnight[4].

Note: When I say that the `hosts` file is idiot-proof and cannot fail in any possible way, I still have to remember that Microsoft has found an innovative way to make it fail nevertheless.

[2]The one that is natively running Linux.

[3]Unfortunately for you if you live in that area of the world where English is considered superior to French, vous devrez vous débrouiller pour traduire l'article dans votre langue.

[4]The risk of the `hosts` file itself suddenly becoming obsolete are, at worst, abysmal; that file became obsolete the day it was designed, and yet every operating system known to mankind has supported it ever since.

I'm just going to write a little[5] program that will run in the background, dynamically updating the `hosts` file depending on the topology of network I'm currently connected to.

Yes but you can't write to that file from an ordinary user!

Of course I know, I've actually read the article I linked two paragraphs back! I will execute my tool as a *Windows Service*. I've never written a Windows Service in my entire life. Good time to start.

A Windows service won't work on a Linux workstation!

Of course I know that also. I will in time make sure that the solution is portable across multiple platforms so that I can reuse most of the decision-making logic, and wrap it inside a thin layer of platform-dependant contraption.

And how will you identify the network you're on?

That is part of the challenge. You can't simply rely on your current IP address, especially if it's in a non-routable range (`192.168.x.x`, `10.x.x.x`, etc.) as these exist everywhere. If my IP address is `192.168.1.23` I could be at home, in a cybercafé or at my great-great-grandcousin-twice-removed log cabin in the woods.

However, if my own IP address is irrelevant, I can still try to scan the network to find recognizable peers. I'm hoping that by probing the attached network cards and trying to identify other equipment connected to the same IP subdomain, my program shall be able to make an educated guess of the identity of the local networks.

For example, if the network has my own printer connected to it, I'm likely at home; if it has a router connected with MAC, address `aa:bb:cc:dd:ee:ff`, I'm likely at my mom's house; if IP geolocation tells me that I'm in the middle of the Appalachian Great Wilderness Reserve, I'm likely at my cousin's cabin.

Especially, I'm quite confident that by querying for the MAC address of the DHCP server, I will be able to make a remarkably reliable guess. Of course I might be blocked from using ARP resolution when I'm at a cybercafé, but I seriously doubt that the Internet connectivity would work if my workstation is unable to find the MAC address of other equipment on the network, and more specifically the default gateway.

Oooh, brainwave! The MAC address of the default gateway might *also* be a clue to the network's identity.

OK, you know where you are. So what?

Now that my program has identified my location, it can use known properties of the available networks to know what to do. If I'm at the office, or if I'm at home with the VPN on, just remove the name mapping from the `hosts` file so that normal DNS resolution can be used. If I'm at home with VPN off, but there is a machine with MAC address `00:11:22:33:44:55` and an IP address on the `192.168.1.x` subnet, and issuing an HTTP get query on the corresponding IP and specific URL returns a valid REST result, then that machine has its *nginx* server running and VPN connected; I can use it as a proxy to access Polarion, so I can map the `polarion` name to that IP address in the `hosts` file.

It makes a hell of a job configuring all those rules, but it will work in the end.

2.2 Remote-Desktopping to my development workstation at the office

For this, I will assume that, even though the problem of determining how to route to the destination has already been already solved the same way the Polarion case was. But I will even assume that I'm willing to have the VPN connected.

You are not going to write a proxy?

No. Again, I already have a solution for that, using an *SSH Tunnel* to forward RDP traffic using my Linux workstation. It works for RDP traffic, albeit there are some strange limitations.

Don't tell me that this too was covered in a previous episode!

[5] I hope you've figured out that this shall not be a *little* program.

Of course it was.

> *That's super easy then: since you know your workstation is named `SLE-DSK-01`, just type that name at the RDP prompt!*

Unfortunately, that does not work. For an unknown reason, my workstation has an IP address in `10.41.0.x` but the DIE DNS server insists on resolving the name to `10.41.32.54`, which was its IP address before the *Big DIE Network Collapse of 2022*. I can safely assume that since then, the DHCP server has changed, and it is no longer talking to the DNS server and DNS name resolution fails.

> *Well, you know what's wrong. Go and tell the IT department about it and let it be.*

I did, but there is no solution yet. I cannot use the DNS name, and I cannot use the IP address because the new DHCP server is issuing much shorter lease times for addresses, so it changes every now and then.

> *And yet its so easy! You write a batch file that dumps the current IP address of your workstation in a text file somewhere on a shared network drives, and you run it once every hour.*

No, it's not that easy. If my workstation just rebooted, I am not logged in, so my scheduled task does not have write access to the network shares.

> *Again, easy! Your task can impersonate you by logging in automatically.*

Oh, you mean make sure to use a technique that is guaranteed to have all IT managers frown at me? Use a technique that guarantees that if I change my password, the service is likely to try again and again to log-in with my old credentials and lock-up my account for good? No thanks.

My project is for the task to regularly determine the current IP address of the machine, and upload it to some external server that I can easily access from the Internet. Hence the name of the project: *The Workstation Status Uploader*.

Then I just have to display an html page in a browser, that page would show me the IP address of my machine, and I can cut-and-paste that address into my RDP prompt.

Oh, and by the way my own workstation is not the only one that cannot be resolved by the DNS. Other people at DIE experience the same issue.

> *You'll have to run the uploader on all those.*

I want to try to avoid that. That would be a deployment nightmare[6]. It's best if I have a single workstation at the office where the uploader works, and it queries all visible network equipment and try to identify what they are.

From the solution to the Polarion issue, I already have the code that scans subnets and identifies what they connect using MAC addresses. I can therefore combine the two solutions in a single service that would identify network appliances, detect what connections are possible, and record the IP address of whatever it finds.

It then uploads all gathered information to the web service, so multiple people can log-in to the service and retrieve what's interesting to them.

> *Whoa! You're going to* ***also*** *develop an online application that receives various information about network equipment, programs and processes, the relations between all those items, all this sent by an application running on a trusted workstation though a REST API, then allows individual users to log-in and obtain the information subset that is relevant to them?*

I'm pretty sure we already have an application like that. Hopefully, it was designed to be open to new kinds of *resources*, *links*, *metrics* and *properties*[7].

[6] And anyway there will always be things connected to the network on which the uploader won't run; I'm pretty sure I won't succeed running it on network printers, Lantronix serial-to-IP converters, or network-based lightpoles; and yet I'd like to retrieve their IP address also.

[7] If, despite all the keywords I just threw at you, you have not by now guessed what application I'm talking about, you need to be more curious about the products your company is developing.

So, instead of patiently waiting for those DNS issues to be addressed, as anyone else in the company, you embark on a journey consisting of writing several thousands of lines of code, a yet-to-be determined amount of Campagnol Chronicles, and a list of new LabX configuration items longer than Manhattan's phonebook, all this while drinking an inordinate amount of strong stale coffee?

Put simply, yes, except that I don't drink coffee, I take Pepsi Max my dear. Anyway, the journey matters more that the destination.

Lao Tseu?

No. Christopher Columbus.

3 What I intend to do in the coming months

I will try to write a program that catalogs the contents of the networks it can reach, consolidates the results and make the topology available to the outside world. I want my program to perform the following tasks:

- Run in the background, without user interaction, as soon as the workstation starts.
- Scan the OS configuration, the reachable networks, etc. in search for identifiable elements such as other workstations, routers, printers, etc. The program must do that often enough to be able to react to changes, but also not too often so that it is not mistaken for a malicious trojan application[8].
- Be as platform-independant as possible. Especially, the application should run on Windows (as a Windows service) and Linux (as a daemon). Linux shall include single-board computers like Raspberry Pis. Mac and OpenBSD support shall be left to the reader as an exercise[9].
- Categorize the trustability of the gathered information; obviously an IP address reported directly by the workstation can be more trusted than one retrieved by scanning the network, itself being more trusted than one returned by a DNS resolution. Consolidate shared information obtained from different sources and build a reliable model of the global configuration.
- Upload the modeled information to online services. A first target shall be the LabX infrastructure, and especially the *resources*, *properties* and *metrics*; this will help ensure we have another client application for the infrastructure, again stressing its openness. In the future, I will try to also target other services as well such as Graphite and Grafana.
- Exchange the models across multiple instances of the application. Each application shall use the received remote models as additional information to refine its own model.
- Analyze the model information to automatically adapt the configuration of the local workstation. Especially, adapt the `hosts` file to mitigate funky DNS resolutions and the need for proxies and gateways. This in turn may require *scripting* so that rules can be adapted by the end user.

As usual, I don't expect anyone else to be interested in actually running this on their own machine.

4 What's next

Now that I disclosed what I want to do, next episodes will take their time to expose how I intend to do all that. I'm pretty sure it will drag on for several months, I just hope I will be done by the time I retire.

[8]Strictly speaking, my program *is* a trojan; it is used for a good cause, but it is nevertheless a trojan.
[9]Unless you offer me a complimentary Macbook Air, of course...