FOSS : A poor man's security analysis architecture.

Parinay Bansal Follow
Sep 21 · 8 min read

To be honest, this post does not cover the entire spectrum of free and open source software for building a security environment for your medium or small enterprise, but makes an effort at the same.

> *The community is a great resource.*

This being my first post, I am trying to bring out my own experience in building my security analysis architecture that I have setup for my own startup.

. . .

A Malware analysis lab

I wanted to build a decent enough platform for Malware and Security analysis experiments.

I started looking at different solutions and products available on the market but to be honest most ideas were shot down for the want of adequate funds required in setting up.

Well that did not bog me down, I decided to go ahead on my own. I pulled up a server grade machine and a few workstation terminals from my IT admin and decided to give it a go.

The server machine had a decent 16 GB of memory and some available storage.

Windows server 2012 and upwards, though a great choice turned out to be an overkill for me. Purchasing a VMWare EXSI would have been expensive for us to afford.

I looked around and found XEN SERVER, a Linux based hypervisor by CISCO which has a lot of features packaged into itself and I instantly knew this is what I was looking for. The community version of Xen Server is free too.

The XEN provided me with 6 VM instances and also gave me a decent testing environment.

. . .

A tryst with FLARE VM

The workstations that my team received, supported virtual box and VMWARE player.

I did not have the wherewithal to go for VMWare Workstation, though it provides great functionality, but I decided to stick with Virtual Box. The base OS on the workstations was Ubuntu 18.04, my all time favourite flavour of Linux.

With Virtual Box fired up, I brought in the developer editions of various windows OS wherein I would test the malware I received.

Microsoft provides a 180 days developer licence for free for most of the OS platforms. They are great for testing.

I fired up a few VMs on virtual box and tweeked them to suit my requirements.

Then I leaned of flare VM, it’s a high powered set of tools curated by Fireye which requires about 30 minutes to setup on a 50 Mbps connection and I decided to give it a try.

> *It’s awesome*

When I first had a look at the tools that have been packed into it, it was a mesmerizing sight.

Anything that I had ever heard of for analysis was already there. For .net it had DNSpy, de4dot, Ghidra for pretty much anything. It also packages tools for reversing Delphi, the new-old malware language. The selection of debuggers including xDBG and WinDbg.

The platform uses the chocolatey framework for getting the software which is the apt / yum on windows. Isn't it cool?

. . .

Setting up the environment

The environment that I needed on the analysis workstations were primarily interconnected VMs with windows running Flare VM, wireshark /tcpdump on the host and remnux.

A workstation would support them all, so I went ahead with it.

The data ingress began and I setup a lab that would test the malwares on these machines, however I collected some data over a period of 3 months and it's management became a problem.

. . .

Building a correlation engine with MISP and CUCKOO

I turned to GitHub to help me.

Tried a few products like Maltego community edition, Yeti platform and a few more, but my faith rests in MISP framework.

The MISP is a Malware threat intelligence, sharing and corelation platform now maintained by The Computer Incident Response Center Luxembourg (CIRCL), and what more it’s built on python and PHP essentially. While I was working on investigations, I found need for certain objects that needed to be incorporated into the framework, and I developed them. The pull requests for the same were immediately accepted and these objects now form part of the master repo. The maintainers for the project are forthcoming and the documentation is awesome too, to get one started.

Having spent some time on analysis so far, I wrote some custom python scripts, with the help of examples provided on the MISP git and I was up an running in no time.

This was the time we had expanded our teams and now I added another org on MISP and the documentation helped me setup the mails and other features such as the threat feeds.

I was also able to write some scripts for automation tasks for MISP which are provided as enrichment.

And in time, we were able to generate a great threat picture for our client.

We then decide to automate some of the basic testing and thus setup a cuckoo sandbox on our servers. Cuckoo with its underlying code in python provides real control and flawlessly integrates with xen and MISP as well. This helps us in correlating the threat picture and provides the necessary time to write Yara rules.

. . .

Migrating to GitLab for secure storage

Having developed faith in us, our first client offered us certain red teaming projects and even code auditing. We went slow there too.

For the red teaming projects, we again went open source and mostly worked with Kali Linux, Commando VM and for data sharing setup an instance of Dradis.

> *Dradis was great for its geeky nature.*

But the clients as always need more and a visualisation lacked. Having developed faith and familiarity with MISP we decided to modify it. Some tweaks here and there, a few scripts and viola, we had a red team data collaboration framework up and running in no time.

The scripts that we make and maintain, have been pushed on GitHub by some of our developers but at times we needed a more private storage. We never bought the paid subscription to GitHub for hosting our private repos but instead fired up an instance of gitlab on our server.

And there we are, we have our own repos available for work and sharing. We were even able to integrate the scripts with our red team collaboration MISP instance for automation with a mere git pull.

. . .

Experimenting CI/CD AND AUTO DEV OPS

> *A code auditing was on the charts next.*

We had to have some teams and clients send us code for review (primarily websites). The process was long drawn, wherein we are required to receive them by mail or certain other media. But then our own gitlab came to rescue again.

We were able to extend our hosted gitlab to our clients and now they can directly push their code on our prem.

We are currently working on integrating our Gitlab instance with Gitlab Runner, dockers and kubernets where we would be able to write test cases for testing our clients code for security using automation.

The idea is to have a setup wherein we can automate the security testing environment, we are working on writing test cases for automation of testing for features such as SQL injection, XSS, IDOR and more on the provided code.

The environment would fire up an instance as soon as the code is submitted, do the basic testing and provide with results which the team can then analyze and shorten the time lapse in environment setup and basic checks.

. . .

Project management with tiaga

As the team sizes increased along with the workload, we would face a problem of project management every now and then. The work flows were earlier managed using Excel, yes the very good old Microsoft Excel. But then it cluttered up quickly and were not able to cope with the deadlines and pipelines. We have a hybrid of waterfall and agile in our working environment among the teams.

We decided to switch to kanban for better management and we would love it in days to come. The white boards were awesome and we wanted to retain that awesomeness. Kanban was perfect and we started our hunt for good on prem kanban solutions.

Someone suggested that we give taiga a shot and it turned out to be perfect. It integrated with our internal Zimbra mail, so did gitlab and

MISP. And there we were.

The features of taiga gave us better management and a lot of control over work flows. We were able to create and manage issues in real time. Assignment and todos integrate into taiga and gitlab which helped us a lot and what was more, it was all for free.

. . .

Communicating with rocket.chat

For communication among the teams, we setup the good old slack channels and integrated the same with our Gitlab, taiga and all good open source software. But it wasn't free. It doesn't support viewing more than 10000 messages so we decided to move onto Rocket Chat.

The software is again open source and freely available from rocket.chat and integration is simple yes elegant. It has its base on Python and Node JS, which makes it awesome by design.

It integrates using Rest API just as slack would do, and it's free.

. . .

Building tutorials and docs

Working with small and large teams, we needed to generate certain amount of documentation and tutorials on the go. Our employees needed to learn a lot of things starting with the use of Gitlab and the git syntax, getting used to typing around the awesome MarkDown format to name a few.

We decided to go in for Gitlab pages. We are currently moving our stuff on gitlab so that our teams can internally setup the tuts and docs for their scripts, software and environment setup.

. . .

Building our own NMS

As we grow older, the leg length increases. We are now experimenting with a lot of open source stuff to setup our very own Network and Asset Management system. With the increasing number of assets that we now have, internal auditing and asset management requirements are growing too.

. . .

Conclusion

When it all is setup and running, I plan to sit on a comfy chair, stare at my server rack and exclaim. I hope that comes up pretty soon.

Open source software is great, and for startups, I would suggest using all things FOSS as it provides you the necessary control and the agility to modify.

We have seen our friends and competition complain on sticking onto a specific product, technology and then not being able to migrate, because the effort of migration is huge and thus their services end up being expensive and at times superfluous, because they don't understand the nitty gritties of the product they use, but one thing that our open source environment taught me, is, understanding the underlying technology of any product or technology that we used, helped us in growing as a nerdy organisation and made sure that the entire team is adequately nerdy, be it management or the work force and there lies our strength.

You can always restart or reset when you have made a mistake, but learning and growing was our motto from the beginning.

Now we know exactly what we are looking for, and if a product is worth the effort and cost, which at times lacks with our competitors.

> *"What's more satisfying, is that we get to give back to the community"*

. . .

Follow Infosec Write-ups for more such awesome write-ups.

InfoSec Write-ups

A collection of write-ups from the best hackers in the world on topics ranging from bug bounties and CTFs to

medium.com