Sujet finale - Cod'INSA 2019

(Voix du mec qui se plaint tout le temps) Oh non, j'ai encore chopé un virus

Architecture

Le sujet de cette année sera mis en oeuvre simplement. Via l'utilisation de Spingboot et de Java, notre jeu se basera sur le concept d'une API RESTful. Cette API permettra aux joueurs, lors de la finale, d'interagir avec le jeu. La communication avec le jeu se fait entre vous et l'API avec le protocole HTTP et le format de donnée JSON.
Vous aurez des informations ainsi que les remontés sur le discord de Cod'INSA situé ici.

Concept

Vous incarnez un virus™ informatique qui se balade librement sur l'Internet mondial. Ce virus tout choupinet a faim, et votre mission sera simple, infecter le plus de machines possibles pour pouvoir vous propager et infecter l'ensemble du réseau mais attention, cela risque de ne pas s'avérer aussi simple que vous le pensez.
Durant votre périple, vous rencontrerez plusieurs types de machines à infecter :

- Le PC de votre grand-mère, mal protégé et vulnérable à souhait. Ce dernier sera facilement infectable.
- Le serveur de PLD MARS, simple, fait perdre 500 ans pour comprendre que le VPN ne pourra pas passer à cause d'une ridicule option à décocher. Ce dernier sera un peu plus difficile à infecter mais vous ouvrira une voie vers l'ensemble du sous-réseau qu'il administre.

Passons maintenant au concret, les mécanismes de jeu :

- Durant le tour, vous pourrez récupérer deux types d'informations :
    - Les machines qui vous sont visibles
    - Les machines de l'ensemble du plateau
- Pendant le tour, vous pourrez soit attaquer, soit déplacer une quantité de code, soit vous défendre face à une attaque extérieure. Chaque action utilise du code malicieux.
- A la fin du tour, les machines que vous avez infectés génèrent du code malicieux.

Il est important de savoir que la phase d'infectation se déroule de la manière suivante :

- Vous indiquez la machine à viser puis, selon la quantité de code utilisé, vous pouvez la fragiliser ou l'infecter totalement. La quantité de code utilisé devra être strictement supérieur à la défense de la machine. Si vous infectez un serveur, les machines associés vous seront plus facile à infecter.
- Une fois infectée, la machine se met à produire du code malicieux pour vous.
- Certaines machines disposent de bonus, les infecter fait gagner le bonus associé et réalise l'action associée.

Le dernier point, mais le point le plus important, vous n'êtes pas seul sur le réseau. Vous avez d'autres virus prêt à tout pour réussir et qui pourront tout comme vous :

- Infecter vos machines pour vous affaiblir.
- Infecter les machines que vous ciblez si vous n'avez pas réussit à les infecter après les avoir attaquées.
- Vous tuer, si elles infectent toutes vos machines.

Le réseau sera un graphe. Le premier a totalement éliminer l'adversaire ou celui qui domine le plus de noeuds à la fin du temps imparti gagne la partie.
Le jeu se déroule par phases/tour durant lesquelles vous pourrez échanger avec le serveur des informations (état du jeu et actions à faire). Ce déroulement sera rythmé par une fin de tour qui interviendra après XXXXXX secondes et qui validera l'ensemble des actions que vous aurez demandé.

Spécifications

Nous vous donnerons l'url pour accéder au .jar associé au sujet sur Discord. En exécutant le .jar, vous lancez un serveur Spring bot dont l'url de notre application est la suivant : http://localhost:8080/
Voici la liste des différents endpoints de notre API RESTful :

- IA/Join : Pour obtenir un token qui vous sera indispensable pour communiquer avec le jeu par la suite.
- Reset : Pour réinitialiser le jeu à l'état de base.
- Start/ChooseMap : Pour changer le plateau du jeu avant de le lancer.
- Start/Game : Pour lancer le jeu (ne peut s'appeler que si au moins deux joueurs ont obtenus un token).
- PlayAction : Pour communiquer vos actions au serveur.
- Get/Board : Pour obtenir l'état du plateau avec l'ensemble de noeuds le constituant.
- Get/Visible : Pour obtenir l'état des noeuds qui vous entourent et leurs propriétaires.
- End/Turn : Pour déclarer que votre IA a fini de jouer pour ce tour-ci.
- Wait : Pour attendre, quand vous avez fini de jouer, et détecter la fin effective du tour actuel.

Voici la liste des bonus présents dans le jeu :

Bonus	Malus
• **(ID 1)** Multiplicateur de débit • **(ID 2)** Multiplicateur des dépenses globales • **(ID 3)** Infestation random (parmis les machines autres que soi)	• **(ID 0)** Gele de la production • **(ID 2)** Multiplicateur des dépenses globales

Voici un tableau récapitulatif des entrées/sorties des différents endpoints :

EndPoint	Verbe HTTP	Header HTTP	Body HTTP	(Exemple) Response HTTP

IA/Join	POST	IAName : string avec le nom de votre IA		{
				"id": "1",
				"status": "success",
				"token": "IAName-XXX"
				}

Reset	GET			{
				"status": "success"
				}

Start/Choose Map	GET	Map : string avec le nom du plateau		{
				"status": "success"
				}

Start/Game	GET			{
				"status": "success"
				}

PlayAction	POST	Token : le token récupéré avec le précédent endpoint.	La liste des actions à réaliser au format JSON tel que :
			
			[{"owner":1,"from":0,"to":1,"qtCode":15},
			{"owner":1,"from":1,"to":0,"qtCode":15}]
			
			où owner correspond à votre id de joueur, from correspond à l'id du noeud de départ du transfert de code, to correspond à l'id du noeud d'arrivée et qtCode correspond à la quantité de code à transférer.
				{
				"status": "success"
				}

Get/Board	GET	Token : le token récupéré avec le précédent endpoint.		{
				"status": "succes",
				"object": {
				"plateau": [
				{
				"id": 0,
				"coordX": 0.5,
				"coordY": 0,
				"production": 5,
				"neighbors": [
				{
				"id": 1
				}],

				"bonus": false,
				"typeBonus": -1,
				"isServer": false
				},{
				"id": 1,
				"coordX": 0.5,
				"coordY": 0.5,
				"production": 10,
				"neighbors": [
				{"id": 0
				},{"id": 2}],
				"bonus": false,
				"typeBonus": -1,
				"isServer": false
				},
				{
				"id": 2,
				"coordX": 0.5,
				"coordY": 1,
				"production": 0,
				"neighbors":
				[{"id": 1}],
				"bonus": false,
				"typeBonus": -1,
				"isServer": false
				}]
				}
				
				}

Get/Visible	GET	Token : le token récupéré avec le précédent endpoint.		{"object": {"visible":[{"id":0, "coordX":0.5, "coordY":0.0, "production":5, "neighbors":[{"id":1}], "bonus":false, "typeBonus":-1, "isServer":false, "owner":1 }, {"id":0, "coordX":0.5, "coordY":0.0, "production":5, "neighbors":[{"id":1}], "bonus":false, "typeBonus":-1,

				"isServer":false, "owner":1}] }, "status":"success"}

End/Turn	POST	Token : le token récupéré avec le précédent endpoint.		{
				"wait": "true",
				"status": "success"
				}
				
				Si la partie se termine, les champs “partyEnd”:”success” et "winner":”IAName” s’ajoutent à la réponse.

Wait	GET	Token : le token récupéré avec le précédent endpoint.		{
				"wait": "true",
				"status": "success"
				}

En cas d’erreur, vous recevrez ce doux et délicieux json :

```
{
        "error": "Un message d’erreur",
        "status":"error"
}
```

Une fois vos IA enregistrées auprès du serveur, faites les attendre sur le end-point Wait et utilisez votre navigateur web pour aller sur localhost:8080/Start/Game.

Vous êtes arrivés à la fin du sujet, codez-nous une IA sympas et éclatez vous !