INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Encadrants

Gildas AVOINE

Barbara KORDY

Glasir

Application au réseau STAR

Rapport de spécifications fonctionnelles

Étudiants

Pierre-Marie AIRIAU

Valentin ESMIEU

Hoel KERVADEC

Maud LERAY

Florent MALLARD

Corentin NICOLE

Résumé

Glasir est le logiciel d'analyse d'arbres d'attaque et de défense que nous allons développer. Pour ce faire, nous allons améliorer ADTool, un outil d'édition d'arbres, avant de l'intégrer à notre logiciel.

Dans ce rapport, nous allons énoncer les limites de l'existant, pour ensuite présenter nos ajouts et nos améliorations. Nous aborderons enfin de manière succincte notre organisation et la planification du projet.

27 novembre 2014

Table des matières

1 Introduction

La sécurisation des systèmes est une problématique majeure de la société moderne. En ce sens, de nombreuses méthodologies ont été développées [3, 2] dans le but d'identifier les risques et de les quantifier. C'est avec cet objectif que le concept d'arbres d'attaque et de défense (« Attack-Defense Trees » en anglais, ou ADTrees) a vu le jour.

Lors de la phase de pré-étude, nous avons pu comprendre l'intérêt pratique de la construction des ADTrees. Leur utilisation permet d'identifier de manière précise les différentes attaques possibles contre un système et de les valuer en termes de coût, de probabilité, etc. ADTool [1] (Attack-Defense Tree Tool), un logiciel développé pour l'implémentation de ces arbres sur support informatique, a été étudié pour ce projet. Lors de sa prise en main, des limites ont été constatées. En effet, dans un cas concret d'expertise en sécurité, le système doit faire face à une multitude d'attaques possibles et, par conséquent, l'ADTree qui les modélisera sera de très grande taille. Dans ce cas, il est difficile pour l'expert d'en extraire des informations pertinentes au premier coup d'œil. Or, ADTool ne fournit pas d'outil permettant à l'utilisateur de simplifier l'analyse de l'arbre.

L'objectif de ce projet est donc la création d'un logiciel intégrant ADTool et permettant de faciliter le travail d'un expert en sécurité, en lui fournissant des outils pour analyser facilement ses ADTrees. Ce logiciel portera le nom de Glasir (prononcé [glaziʁ]). Il s'agit du nom d'un arbre aux feuilles d'or dans la mythologie nordique [4].

Ce rapport présente les spécifications fonctionnelles de Glasir. Tout d'abord, les limites d'ADTool seront abordées, afin de justifier l'intérêt de Glasir. Puis les différentes fonctionnalités destinées à l'analyse des ADTrees seront décrites. Enfin, quelques améliorations supplémentaires seront également précisées pour offrir un meilleur confort de création et d'édition d'arbres. Ces spécifications seront faites en prenant en exemple une situation précise : celle d'un expert en sécurité chargé par le Service des Transports en commun de l'Agglomération Rennaise (STAR) de déterminer les failles de leurs systèmes de paiement.

2 ADTool : étude de l'existant

ADTool est un logiciel libre permettant aux utilisateurs de modéliser des scénarios d'attaque et de défense sous la forme d'ADTrees. Il est disponible en téléchargement sur Internet (avec ou sans ses dépendances), et il existe aussi une version utilisable directement en ligne. Dans le but de réaliser une solution d'aide aux experts en sécurité modélisant les ADTrees, il nous a semblé cohérent de commencer par étudier l'existant. ADTool est le seul logiciel libre disponible implémentant les ADTrees : cette section détaillera ses atouts et ses points faibles.

ADTool permet de construire des ADTrees sans difficulté. Le logiciel offre la possibilité de créer de nouveaux nœuds, d'éditer leurs labels et de leur ajouter des fils de façon simple et intuitive. Cette dernière fonctionnalité correspond à l'action « Add Child » visible sur la FIGURE 1. Sur cette même figure, il est montré que les différentes actions s'effectuent de façon pratique par le biais de la souris ou de raccourcis clavier. Il est aussi possible d'ajouter des frères à un nœud choisi. Le choix des opérateurs entre ces nœuds (conjonction ou disjonction) se fait rapidement, et la mise en place de défenses est également très facile à réaliser.

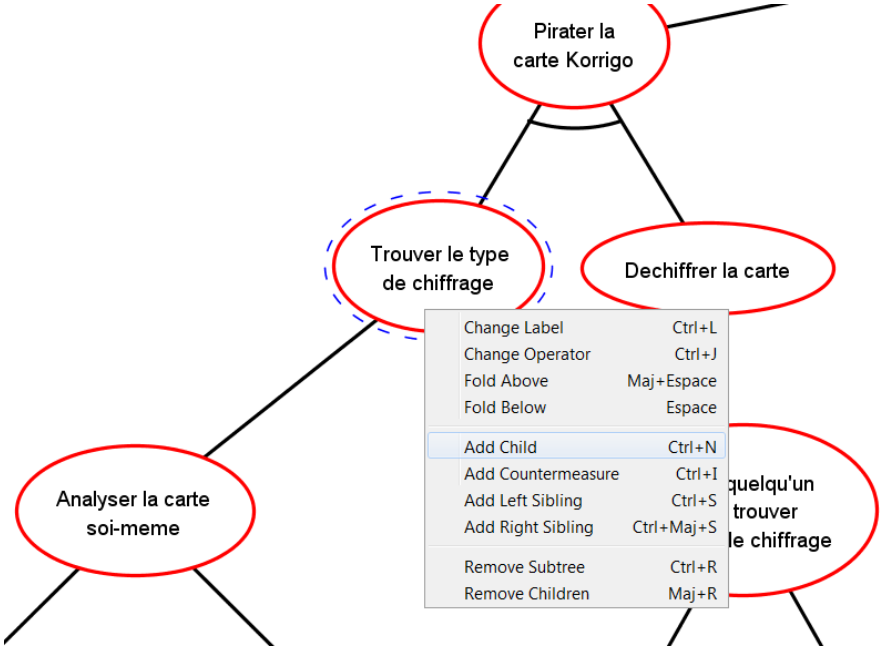

FIGURE 1 – La création d'un arbre dans ADTool est simplifiée par des actions accessibles facilement.

Une fois l'arbre établi, il est possible d'ajouter des valuations à chaque feuille [1] de l'arbre selon un paramètre, qu'ADTool se charge ensuite de propager aux nœuds pères de façon récursive jusqu'à la racine. ADTool dispose à l'heure actuelle de treize paramètres de base (appelés *domains*), présentés dans la TABLE 1.

Dans l'exemple de la FIGURE 2, chaque nœud possède une valuation selon le paramètre « difficulté de réalisation ». Comme indiqué dans la TABLE 1, cette valuation peut prendre les valeurs L, M, H ou E. En éditant la difficulté de réalisation de la feuille « Essayer les clés de chiffrage possibles » à *Medium*, la valuation du nœud père « Casser le chiffrage » est mise à jour automatiquement pour prendre la valeur *Medium*. En effet, la difficulté de ce nœud conjonctif correspond à

1. Une feuille est un nœud n'ayant aucun fils.

Paramètre	**Valeurs possibles**
Difficulty for the proponent (L,M,H)	Low (bas), Medium (moyen), High (élevé) ou l'infini.
Difficulty for the proponent (L,M,H,E)	Low (bas), Medium (moyen), High (élevé), Extreme (extrême) ou l'infini.
Minimal cost for the proponent (not reusable)	Valeurs réelles positives, ou l'infini.
Minimal skill level needed for the proponent	Valeurs entières positives, ou l'infini.
Minimal time for the proponent (in parallel)	Valeurs réelles positives, ou l'infini.
Minimal time for the proponent (sequential) (*temps minimal séquentiel*)	Valeurs réelles positives, ou l'infini.
Overall maximal power consumption	Valeurs réelles positives, ou l'infini.
Probability of success	Valeurs réelles entre 0 et 1.
Reachability of the proponent's goal in less than k units (in parallel)	Valeurs entières de 0 à k.
Reachability of the proponent's goal in less than k units (sequential)	Valeurs entières de 0 à k.
Satisfiability for the opponent	Vrai ou faux.
Satisfiability for the proponent	Vrai ou faux.
Satisfiability of the scenario	Vrai ou faux.

TABLE 1 – Description des paramètres de base d'ADTool.

la difficulté la plus élevée parmi celles de ses fils. Puis, à son tour, la valuation du père de ce nœud père est recalculée à *Medium* car, étant un nœud disjonctif, sa difficulté de réalisation correspond à la difficulté la plus faible parmi celles attribuées à ses fils. L'utilisateur n'a donc pas de calcul à faire lui-même pour obtenir la valuation de la racine de son arbre, c'est-à-dire la valuation de son objectif final.

ADTool automatise donc la création des ADTrees et leur valuation. Mais cette étude met également en avant plusieurs limitations quant aux possibilités d'analyse pour l'utilisateur, dont voici les deux plus importantes :

Exploitation des valuations L'utilisateur n'a aucun moyen de déterminer automatiquement le « meilleur chemin » permettant d'atteindre la racine de l'arbre selon un paramètre donné. Il doit le trouver manuellement, en analysant la façon dont les valuations se sont propagées. Sachant qu'un arbre atteint très rapidement une taille conséquente, cette opération peut prendre beaucoup de temps et représenter un obstacle pour l'expert.

Affichage des paramètres L'analyse de l'arbre est également restreinte par le fait que la valuation d'un nœud (et donc de l'arbre entier) ne peut se faire que selon un seul paramètre à la fois. Il n'est pas possible d'afficher à la fois la « difficulté » et le « temps nécessaire » à la réalisation pour un nœud donné, par exemple. Ceci est contraignant car il est parfois difficile de séparer ces deux notions : par exemple, le nœud « Essayer les clés de chiffrage possibles » est très facile à effectuer (il suffit de taper les clés une à une jusqu'à trouver la bonne), mais le temps nécessaire est colossal.

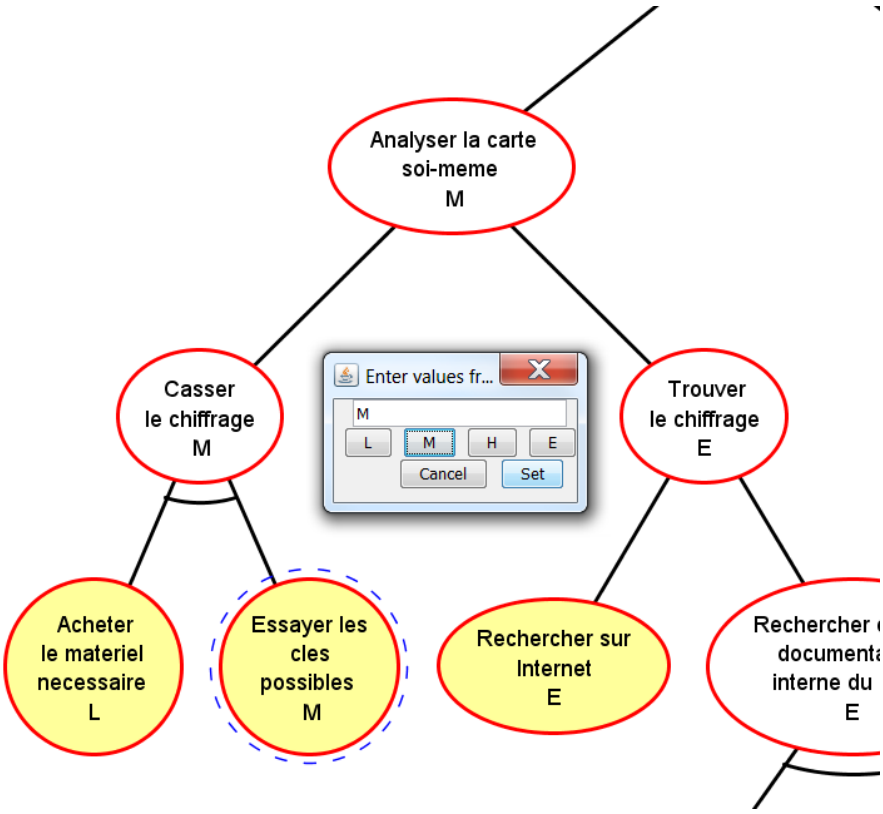

FIGURE 2 – L'ajout de valuations aux feuilles est simple. Ces valuations se propagent ensuite récursivement à tous les antécédents.

ADTool ne permet donc pas d'obtenir une vision complète de l'arbre par combinaison de plusieurs paramètres.

Les deux lacunes précédentes ont été mises en avant car elles rendent l'analyse des arbres édités laborieuse. La prochaine section détaillera l'architecture du logiciel Glasir, créé dans le but de dépasser les limitations évoquées.

3 Architecture logicielle

L'étude d'ADTool a mis en évidence des limitations en ce qui concerne l'analyse des arbres. Afin d'y remédier, de nouvelles fonctionnalités seront implémentées. Ces dernières se décomposent en deux parties : celles qui apportent une véritable plus-value pour l'analyse des arbres, et celles qui facilitent le travail d'édition d'arbres.

Le projet aboutira donc sur la création d'un nouveau logiciel nommé Glasir qui contiendra les nouvelles fonctionnalités d'analyse. Il encapsulera une version améliorée d'ADTool, qui sera son éditeur d'ADTrees. Deux raisons ont motivé la décision de créer un nouveau logiciel. Tout d'abord, cela permet de séparer l'analyse et l'édition des ADTrees, afin d'avoir des logiciels dédiés à leur tâche. Deuxièmement, cette solution permet d'utiliser des technologies différentes de celles d'ADTool, enrichissant ainsi notre formation. La FIGURE 3 montre comment ADTool sera intégré

dans Glasir, ainsi que les modules qui seront développés.

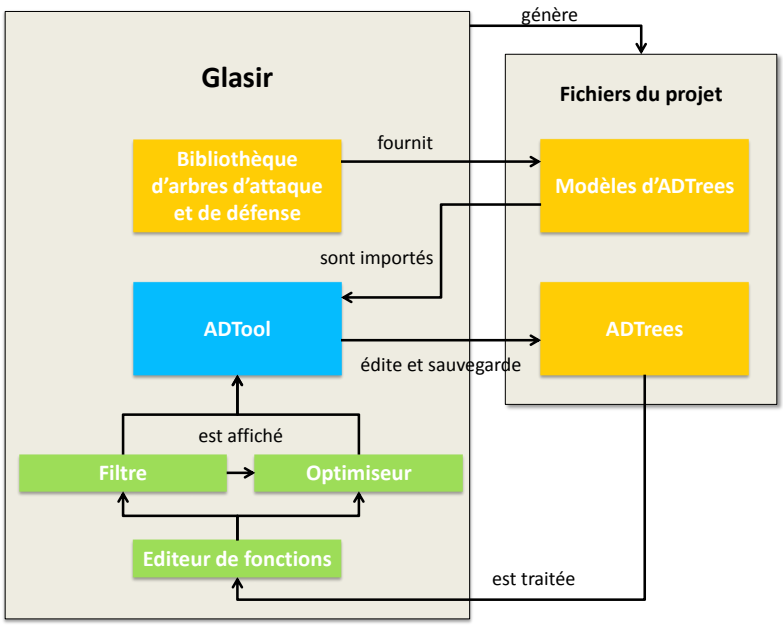

FIGURE 3 – Architecture du logiciel Glasir.

Comme illustré sur la FIGURE 3, Glasir est constitué d'une bibliothèque d'ADTrees, d'ADTool et des nouvelles fonctionnalités d'analyse. L'analyse réalisée par l'expert en sécurité sera sauvegardée sous forme de projet. Celui ci contiendra les ADTrees de l'analyse ainsi que les modèles génériques d'ADTrees que l'expert a jugés utiles. Ceux-ci proviennent de la bibliothèque d'ADTrees de Glasir.

Lors de la création d'un projet quelconque, Glasir commence par créer les fichier de base d'un projet. L'expert va alors créer ou modifier ses ADTrees grâce à ADTool. Les modèles de la bibliothèque d'ADTrees utilisés dans le cadre du projet seront également sauvegardés dans le projet, pour gérer leurs éventuelles modifications sans impacter la bibliothèque d'arbres génériques de Glasir. L'expert pourra alors définir ses paramètres de synthèse avec l'*Éditeur de fonctions*, filtrer les arbres grâce au module *Filtre* ou encore y chercher le chemin optimal à l'aide de l'*Optimiseur*. La prochaine section détaillera l'implémentation de ces différents modules.

4 Fonctions d'analyse

Comme précisé dans la SECTION 3, Glasir est constitué de trois modules principaux pour atteindre son objectif d'aide aux experts en sécurité. Ces modules proposent de nouvelles fonctionnalités d'analyse des ADTrees qui seront détaillées dans cette section.

4.1 Optimiseur

Une fois que l'expert a obtenu son arbre complet avec ADTool (qui peut se composer de plusieurs milliers de nœuds), il ne peut pas facilement identifier le chemin optimal selon un paramètre donné. Il s'agit en effet d'un travail manuel, relativement fastidieux, qui doit être recommencé à chaque modification de l'arbre.

Pourtant, la méthode utilisée est systématique. Il suffit de parcourir l'arbre en partant de la racine et de sélectionner à chaque étage le nœud le plus intéressant. Pouvoir identifier automatiquement le chemin optimal fera gagner beaucoup de temps à l'expert, et permettra aussi de limiter les risques d'erreurs. Ce processus peut donc être implémenté dans Glasir.

Nous allons donc créer un *optimiseur* afin de répondre à ce besoin. Ses entrées seront les suivantes :

- un arbre provenant du projet ;
- le paramètre à prendre en compte.

En sortie sera obtenu un nouvel arbre, sous-graphe de l'arbre d'entrée, contenant le chemin optimal. L'expert pourra ensuite le traiter comme un tout nouvel arbre, en fonction de ses besoins.

Prenons en exemple l'arbre de la FIGURE 5. Si l'objectif est de trouver un chemin optimal pour le paramètre « coût minimal », le résultat est l'arbre de la FIGURE 4.

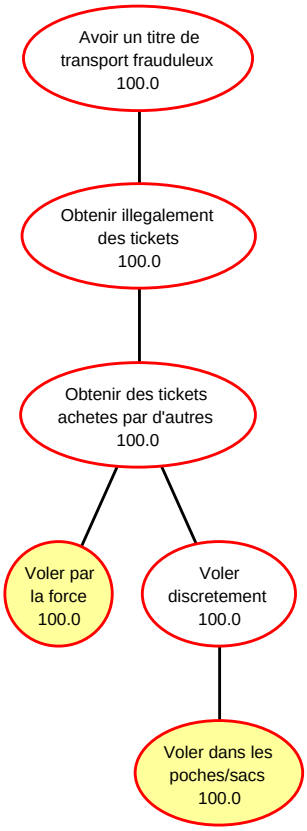

FIGURE 4 – L'attaque optimale (par rapport au coût minimal) est ainsi facilement lisible.

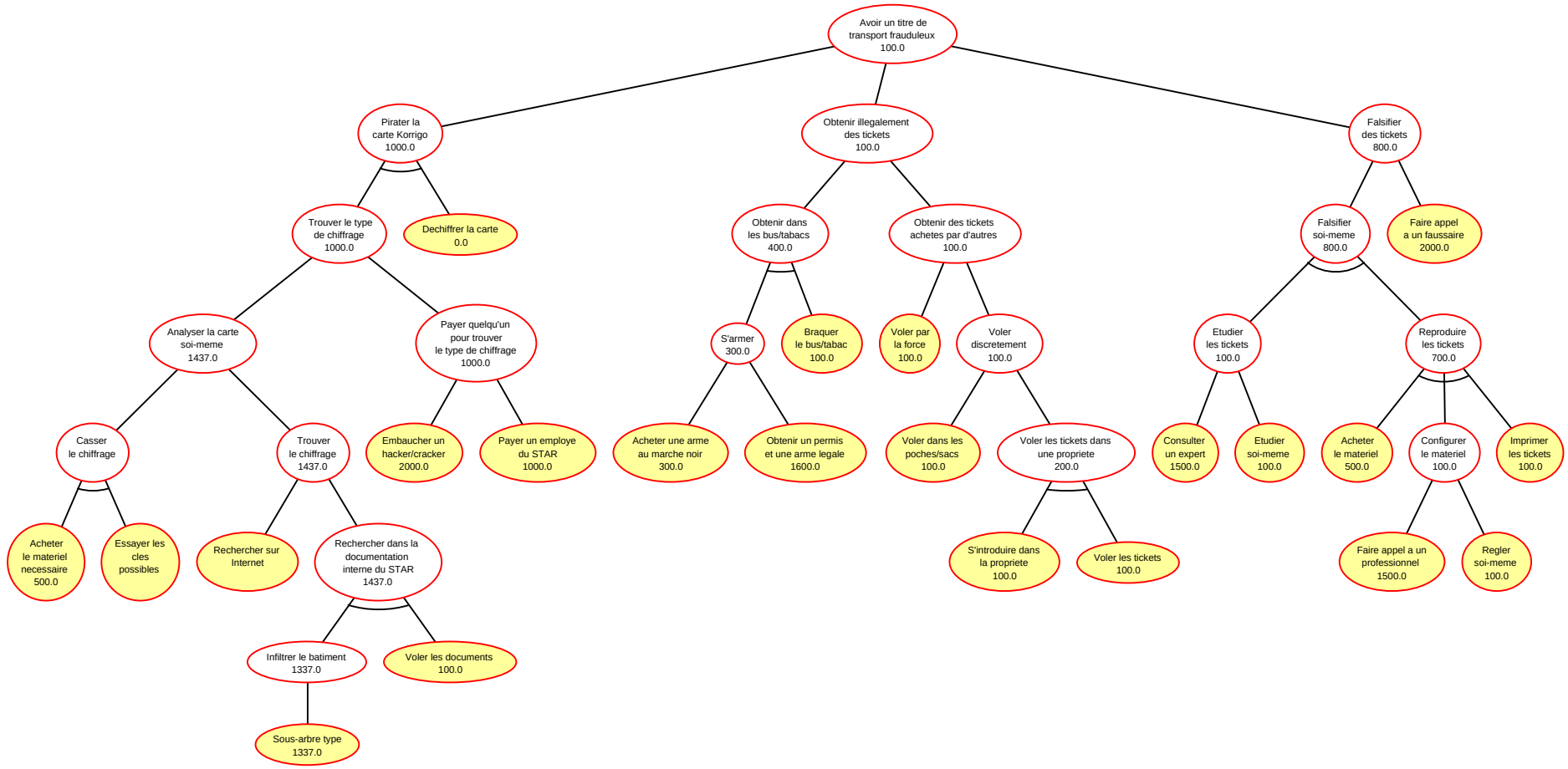

FIGURE 5 – L'arbre avant l'utilisation de l'optimiseur.

L'algorithme parcourant l'arbre à la recherche du chemin idéal est détaillé dans l'Algorithme 1. Les notations non explicites sont présentées ci-dessous :

Algorithm 1 opti(racine, param)

```
l_fils = fils(racine)
if vide(l_fils) then
  return
end if

if mode(racine) == ou then
  v = param(racine)
  for n in l_fils do
    if not defense(n) and param(n) != v then
      delete(n) // will delete subtrees as well
    end if
  end for
end if

for n in fils(racine) do
  opti(n, param)
end for
```

- `racine` correspond au nœud à partir duquel l'arbre sera élagué ;
- `param` est une fonction renvoyant une valeur pour un nœud donné ;
- `fils` est une fonction renvoyant la liste des fils du nœud passé en paramètre ;
- `mode` précise si il s'agit d'un nœud disjonctif ou conjonctif ;
- `defense` est une fonction prenant un nœud en entrée et renvoyant un booléen indiquant si il s'agit d'un nœud d'attaque ou de défense.

L'algorithme est récursif et modifie l'arbre en l'état (il est donc obligatoire de travailler sur une copie de l'arbre). Ainsi, pour lancer l'optimisation, la fonction `opti` sera appelée avec la racine de l'arbre en paramètre (il s'agit donc d'un parcours dit « top-down ») . Dans le pire des cas, tous les nœuds seront à garder, et la complexité sera donc en $\mathcal{O}(n)$, n étant le nombre de nœuds de l'arbre.

4.2 Filtre

Souvent, l'expert en sécurité va chercher à se défendre contre un attaquant précis. Dans ce cas, il va identifier les ressources dont l'attaquant dispose (temps, argent, etc.), pour ne conserver que les chemins de l'arbre empruntables par l'attaquant.

Nous allons donc implémenter une fonctionnalité de *filtre* permettant de répondre à ce besoin. L'expert devra définir un critère selon lequel effectuer le filtrage, puis choisir un intervalle dans lequel les valeurs du paramètre devront se situer. Il sera possible de filtrer l'arbre selon plusieurs paramètres à la fois.

Les entrées de la fonction de filtrage seront donc les suivantes :
- l'arbre à filtrer ;
- les valuations de l'arbre servant de critères pour le filtrage ;
- les intervalles de sélection sur les différentes valuations.

L'arbre retourné par la fonction de filtrage sera une copie élaguée de l'arbre original. Seuls les chemins respectant tous les intervalles de filtrage indiqués seront conservés.

Par exemple, en appliquant un filtre sur l'intervalle $[0, 500]$ et sur la valuation « coût minimal » à l'arbre de la FIGURE 5, l'arbre obtenu est celui de la FIGURE 6. En effet, il contient uniquement les valeurs comprises dans l'intervalle spécifié. L'arbre élagué par le filtre a été considérablement

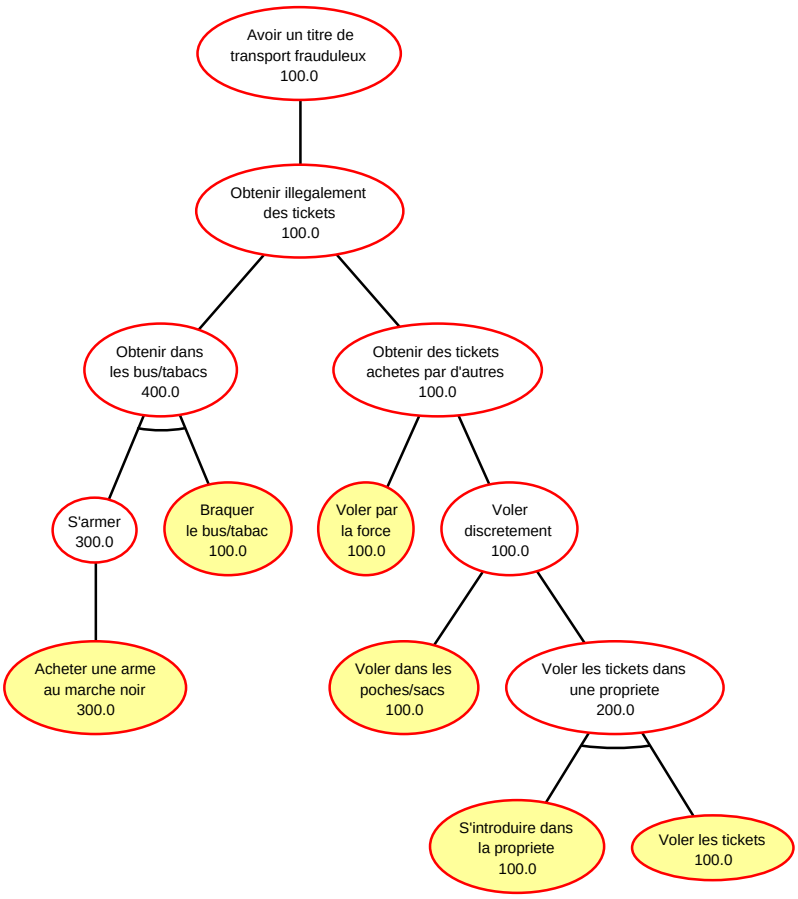

FIGURE 6 – Arbre filtré selon le coût minimal pour l'attaquant.

réduit. L'arbre exemple étant de très petite taille comparé aux modélisations de systèmes réels, cette fonctionnalité sera d'une grande aide pour l'expert.

Nous utiliserons dans ce module l'algorithme 2. Cet algorithme est lui aussi récursif, et son nombre d'appels sera au pire en $\mathcal{O}(n)$ (n étant toujours le nombre de nœuds de l'arbre).

- `racine` est le point de départ de l'algorithme ;
- `rules` est l'ensemble des règles de filtrage (valuations et intervalles).

4.3 Éditeur de fonctions

L'expert est actuellement cantonné aux paramètres présentés dans la SECTION 2, et ne peut pas en créer d'autres, ce qui limite ses possibilités d'analyse. En effet, les paramètres déjà existants ne sont pas les seuls pouvant intéresser un expert en sécurité. Nous souhaitons donc rendre possible la

Algorithm 2 filtre(racine, rules)

```
for r in rules do
  if not r(racine) then
    delete(r) // will delete subtrees as well
    return
  end if
end for

for n in fils(racine) do
  filtre(n, rules)
end for
```

création de nouveaux paramètres, à partir de ceux déjà disponibles et des fonctions mathématiques basiques (division, multiplication, min, max, etc.). Ces nouvelles valuations pourront ensuite être appliquées à n'importe quel arbre, de la même manière que les paramètres de base. Elles pourront ainsi être utilisées pour évaluer les arbres selon de nouveaux critères, et si besoin pour les élaguer à l'aide du filtre que nous allons implémenter.

Cette nouvelle fonctionnalité prendra donc en entrée les éléments suivants :
- les paramètres intervenant dans la synthèse ;
- les opérations mathématiques appliquées ;
- le nom de la fonction générée.

Par exemple, dans l'arbre de la FIGURE 5, les deux sous-arbres « Obtenir illégalement des tickets » (sous-arbre A) et « Falsifier des tickets » (sous-arbre B) permettent d'atteindre l'objectif final. Il est actuellement possible de les valuer par treize paramètres, nous n'en garderons ici que deux pour simplifier : le coût minimal et la probabilité de succès.

On constate rapidement que choisir A implique un coût de réalisation moindre pour l'attaquant, puisqu'il nécessite peu de matériel. Cependant, un vol est toujours risqué, et les chances de se faire arrêter sont élevées, diminuant fortement la probabilité de succès. Opter pour le sous-arbre B, bien que plus cher à réaliser, permet de limiter ce risque et d'augmenter la probabilité de succès. On voit donc que ces deux paramètres, qui ne sont pas liés, peuvent être tous les deux utiles à l'expert et entraîner deux interprétations très différentes. C'est pourquoi il serait intéressant de pouvoir les combiner, grâce à notre éditeur de fonctions, afin de les prendre en compte simultanément. C'est ensuite à l'expert de juger des liens qu'il désire instaurer entre les paramètres : fonctions mathématiques, coefficients, etc. Par exemple, s'il décide d'appeler le nouveau paramètre *result*, et qu'il estime qu'il s'agit du coût auquel on ajoute la probabilité de succès multipliée par deux, on obtient la synthèse suivante :

$$result = (minimal\_cost) + 2 \times (probability\_of\_success).$$

La synthèse *result* sera désormais disponible pour donner des valuations aux nœuds de n'importe quel arbre, et pourra elle-même être utilisée pour créer d'autres paramètres.

Les principaux modules de Glasir étant présentés, la prochaine section détaillera l'implémentation des différentes fonctionnalités dans Glasir destinées à le rendre simple et ergonomique. De plus, elle traitera des améliorations apportées à ADTool pour le rendre plus pratique pour l'utilisateur.

5 Ergonomie du logiciel

Des fonctionnalités plus secondaires seront implémentées dans le but de rendre Glasir suffisamment ergonomique pour être utilisé avec confort. Ces fonctionnalités n'apporteront pas de nouveautés en terme d'analyse des ADTrees, mais permettront de faciliter la manipulation des arbres dans ADTool et de rendre Glasir simple d'utilisation.

5.1 Ouverture simultanée de plusieurs arbres

Dans sa version actuelle, ADTool ne permet de travailler que sur un seul arbre à la fois. Cette limitation est assez handicapante : par exemple, l'utilisateur ne peut pas ouvrir deux arbres en même temps pour les comparer. Glasir permettra donc d'ouvrir plusieurs instances d'ADTool, chacune contenant un arbre. Ces différentes instances se regrouperont sous la forme d'onglets situés dans l'interface de Glasir.

5.2 Hiérarchie des arbres

L'ouverture simultanée de plusieurs arbres induit la possibilité de manipuler un grand nombre d'arbres dans le cadre d'un même projet. Dans le but d'aider l'utilisateur à organiser facilement son projet, Glasir fournira dans un dock latéral une arborescence de dossiers et de sous-dossiers contenant les différents arbres utilisés.

5.3 Bibliothèque de modèles

Il n'est pas forcément facile pour l'utilisateur de créer un arbre en partant de zéro. C'est pourquoi Glasir fournira une bibliothèque d'arbres génériques, aidant ainsi l'expert à démarrer son projet. Cette bibliothèque de modèles sera dupliquée pour être propre à chaque nouveau projet, permettant à l'utilisateur de modifier, compléter ou élaguer les arbres selon ses besoins sans impacter les originaux. Notre étude portant sur le réseau STAR, la bibliothèque initiale de modèles de Glasir intégrera principalement des ADTrees génériques relatifs aux réseaux de transport en commun.

Si l'utilisateur construit un nouvel arbre qu'il juge pertinent de réutiliser dans un autre projet, il pourra enregistrer cet arbre comme nouveau modèle, qui se rajoutera donc à la bibliothèque. Glasir verra ainsi sa bibliothèque de modèles s'étoffer progressivement au cours du temps.

5.4 Couper/copier/coller

Actuellement, ADTool ne permet pas l'utilisation des fonctions habituelles *couper*/*copier*/*coller*, qui pourraient pourtant s'avérer pratiques lors de la création d'un arbre. Par exemple, dans le cas de l'oubli d'un nœud père, l'instauration de ces fonctionnalités permettrait de déplacer facilement les fils concernés de l'ancien nœud père vers le nouveau.

En conséquence, ADTool sera modifié pour implémenter ces fonctionnalités. Ainsi, la sélection d'un nœud entraînera la sélection de ses fils, afin de pouvoir couper ou copier ces nœuds facilement.

Les raccourcis clavier « classiques » de ces fonctions (*CTRL+X, CTRL+C, CTRL+V*) seront mis en place.

5.5 Amélioration du codage des arbres

ADTool affiche actuellement dans son interface une section nommée *ADTerm Edit*. Celle-ci contient une représentation de l'arbre sous un format texte, en utilisant un langage propre au logiciel. Lors de la modification de l'arbre dans l'éditeur graphique, ADTool met à jour le texte correspondant en temps réel et de manière automatique. L'inverse est également vrai : il est possible de changer les labels des nœuds, ou les opérateurs, directement depuis *ADTerm Edit* afin d'afficher ensuite le résultat graphiquement.

Mais le langage qui permet de décrire les arbres sous format texte n'est pas très lisible : il ne contient par exemple pas le nom des nœuds autres que les feuilles. Sur la FIGURE 7, le code indique le nom des deux feuilles « Acheter le matériel nécessaire » et « Essayer les clés possibles », et la conjonction est bien précisée par l'opérateur « ap » (« op » en cas de disjonction). Mais aucune référence n'est faite au label du nœud père « Casser le chiffrage ». La grammaire utilisée sera modifiée pour corriger ce défaut, afin de rendre l'utilisation de cette fenêtre plus intuitive.

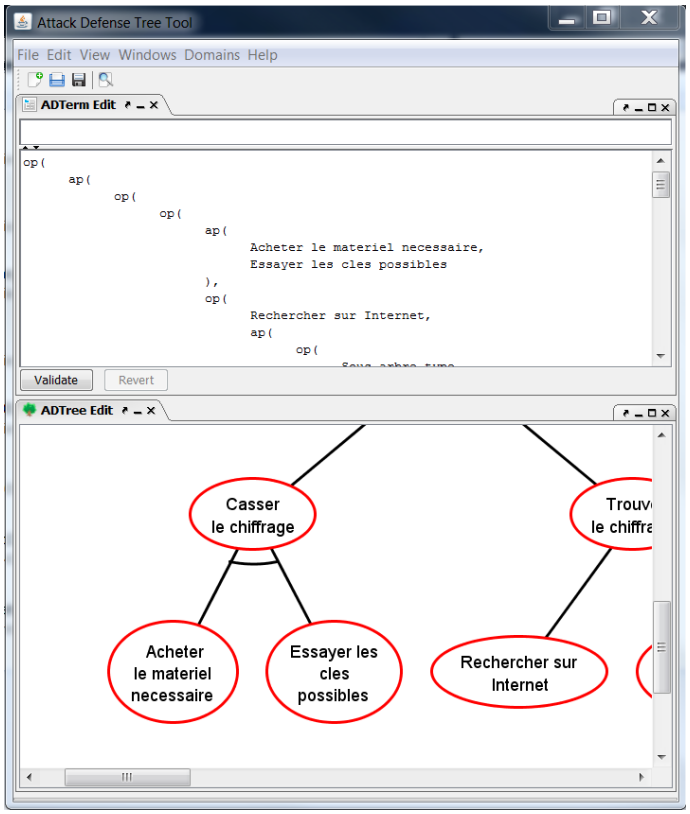

FIGURE 7 – L'interface d'ADTool. En haut, l'arbre au format texte ; en bas, sa représentation graphique.

5.6 Annulation d'une action

Pour le moment, effectuer une action sous ADTool est irréversible. Cela n'est pas grave pour certaines actions rapides à effectuer, telles que le renommage d'un nœud. Mais supprimer un nœud (ce qui implique la disparition de tous ses nœuds fils s'il y en a) par erreur peut entraîner un travail énorme, et donc une perte de temps. La possibilité de revenir à l'état précédent de l'arbre (par le raccourci clavier classique *CTRL+Z*) éviterait d'avoir à refaire ce travail de construction parfois fastidieux. Nous souhaitons créer au moins une sauvegarde de l'état précédent, afin de pouvoir annuler la dernière modification effectuée. Si possible, nous envisageons l'implémentation d'une pile circulaire contenant les N dernières actions (chaque modification entraînant la création d'un nouvel état) avec un curseur pointant sur l'état courant. Cela permettrait de revenir en arrière sans contraintes.

Cette fonctionnalité sera implémentée en utilisant le patron de conception *Commande*. Les actions de l'utilisateur dans ADTool seront sauvegardées pour pouvoir, quand l'utilisateur demandera l'annulation de son action, effectuer l'action inverse.

5.7 Vue globale des paramètres

Dans l'état actuel d'ADTool, lorsqu'un arbre est valué, un seul paramètre est affiché (le même pour tous les nœuds) même si plusieurs paramètres sont utilisés en réalité. En effet, chaque valuation a un onglet propre, dans lequel elle est la seule effectivement affichée sur l'arbre. Nous souhaitons créer un onglet plus général, dans lequel tous les paramètres voulus peuvent être visibles sur l'arbre en simultané. C'est ensuite à l'expert de décider des paramètres qu'il juge utile d'afficher. Cela serait aussi valable pour les fonctions évoquées dans la Sous-section 4.3.

Pour une meilleure lisibilité, chaque paramètre aura une couleur différente, et l'arbre sera accompagné d'une légende résumant ce jeu de coloration et ses correspondances. Nous devrons également gérer la taille des nœuds, pour que les paramètres ne dépassent pas de la bulle. Un tel système étant déjà présent dans ADTool pour gérer les labels, il suffira de l'étendre à l'affichage des paramètres.

Maintenant que toutes les fonctionnalités à développer ont été définies, une première planification peut être présentée.

6 Organisation

Cette planification sera le sujet d'un prochain rapport en décembre, mais nous pouvons déjà en donner une ébauche. Celle-ci sera entre autres illustrée par une chronologie MS Project.

6.1 Versions

Nous avons découpé le développement de Glasir en différentes versions, chacune incrémentale en fonctionnalités. Cela nous permettra de toujours avoir un produit fonctionnel, et de faciliter les phases de tests.

Version	Description
0.1	Application ouvrant ADTool
0.2	Création d'un projet
0.3	Paramètre de synthèse
0.4	Filtre
0.5	Optimiseur
0.6	Bibliothèque de modèles
1.0	Version finale

TABLE 2 – Tableau énumérant les différentes versions de Glasir.

Toutes les améliorations sur ADTool seront réalisées au fur et à mesure, en parallèle du développement de Glasir.

6.2 Planification

Une ébauche de la planification se trouve à la FIGURE 8. Cette dernière donne un aperçu du temps de développement qui sera consacré à chaque version de Glasir.

6.3 Répartition des tâches

Au second semestre, seuls resteront Pierre-Marie AIRIAU, Valentin ESMIEU et Maud LERAY à travailler sur ce projet, étant donné que le reste du groupe part étudier à l'étranger. Par conséquent, nous comptons séparer le travail ainsi :

- l'un d'entre nous développera les fonctionnalités d'analyse de Glasir à proprement parler ;
- le deuxième se tournera plutôt vers l'intégration d'ADTool dans Glasir ;
- le troisième apportera les modifications à ADTool.

Valentin suivant le deuxième semestre de la troisième année à partir de mi-janvier, cette répartition sera problablement affectée par les dates d'examens qui ne concordent pas entre la troisième et quatrième année.

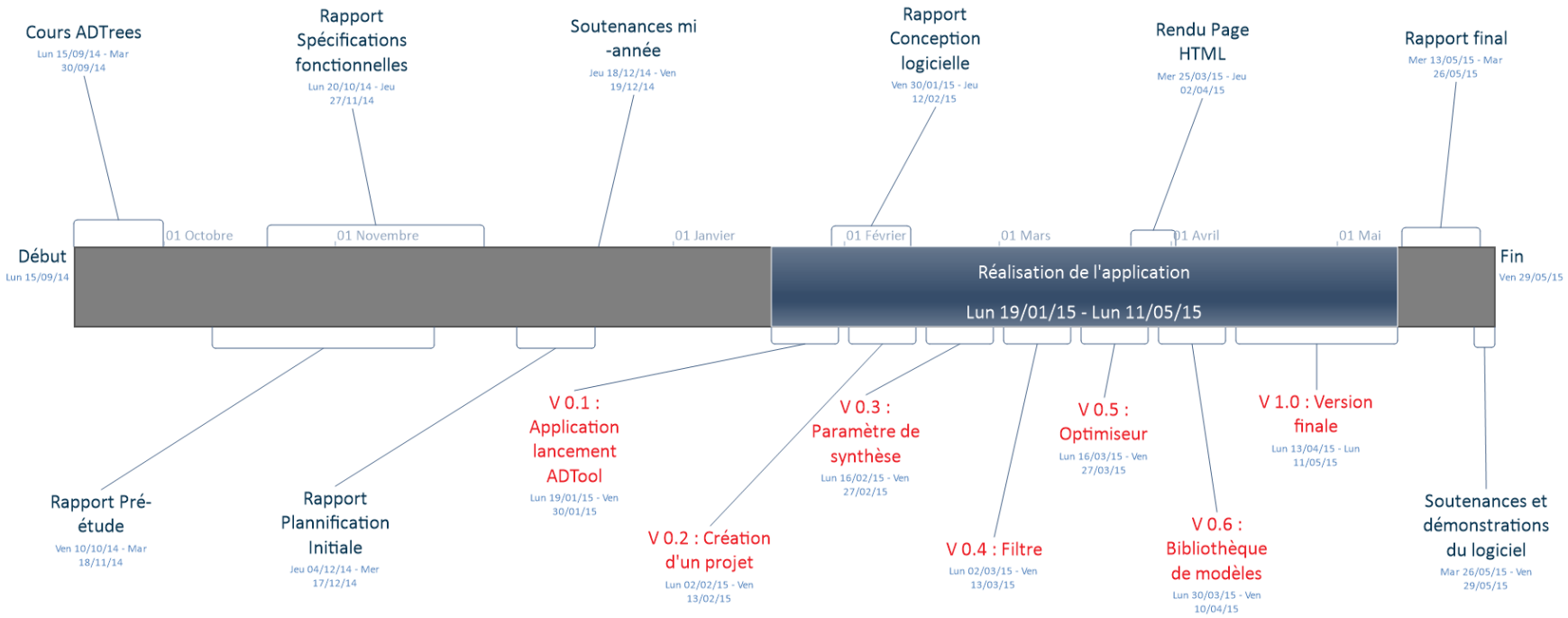

FIGURE 8 – Planification modélisée sous MS Project.

7 Rétrospective et conclusion

Rétrospective Suite à la rédaction du rapport précédent (pré-étude), nous avions constaté que notre organisation de travail était à revoir. En effet, nous avions trop cloisonné la rédaction des différentes parties, et il nous manquait un superviseur chargé d'harmoniser le tout. Le résultat fut un rapport dont les parties ne s'enchaînaient pas toujours correctement, et avec des styles de rédaction très variables : cela manquait d'homogénéité.

De plus, nous avions trop tardé à faire nos « brainstorming » pour déterminer précisément ce que nous allions écrire, ce qui a grandement retardé la rédaction. Nous avions même dû supprimer une partie de notre travail.

À partir de ces constats, nous avons décidé de démarrer ce deuxième rapport par une série de réunions. Cela nous a permis de commencer la rédaction en ayant une idée précise de ce que nous allions écrire. Cette fois, nous avons désigné un responsable d'harmonisation, Corentin Nicole, qui a moins rédigé mais dont le rôle était de s'assurer de la cohérence globale du rapport tout au long de son avancement.

Conclusion Après avoir présenté les limites du logiciel existant, ADtool, ce rapport a détaillé l'implémentation des principaux modules du logiciel Glasir. Celui-ci, développé essentiellement pour ce projet, permettra de dépasser les limitations d'ADTool grâce à trois fonctionnalités principales : l'éditeur de fonctions, l'optimiseur et le filtre. Ces nouvelles fonctionnalités visent à améliorer les possibilités d'analyse des arbres pour l'expert en sécurité. Glasir encapsulera ADTool et l'utilisera pour réaliser la création et l'édition des ADTrees.

Dans un second temps, ce rapport a présenté les modifications qui seront apportées à ADTool dans le cadre de ce projet. Des fonctionnalités seront implémentées pour faciliter l'édition d'arbres. La possibilité d'effectuer un copier/coller, une annulation d'action ou encore d'ouvrir plusieurs arbres à la fois sont des exemples de fonctionnalités usuelles qui manquent à ADTool et qui y seront ajoutées.

Ce rapport précise donc le cahier des charges précédemment établi. Il détaille l'architecture du logiciel Glasir, avec ses modules principaux et leurs interactions. La prochaine étape sera la planification précise du travail d'implémentation à effectuer.

Références

[1] Barbara Kordy, Piotr Kordy, Sjouke Mauw, and Patrick Schweitzer. ADTool : Security Analysis with Attack–Defense Trees. In Kaustubh R. Joshi, Markus Siegle, Mariëlle Stoelinga, and Pedro R. D'Argenio, editors, *QEST*, volume 8054 of *LNCS*, pages 173–176. Springer, 2013.

[2] Barbara Kordy, Ludovic Piètre-Cambacédès, and Patrick Schweitzer. DAG-Based Attack and Defense Modeling : Don't Miss the Forest for the Attack Trees, 2014. DOI : 10.1016/j.cosrev.2014.07.001.

[3] Jean Leneutre. Concepts fondamentaux de la sécurité. `http://www.infres.enst.fr/people/leneutre/SECUR/INF941-Intro-securite-2011-12.pdf`, 2014. [Online ; accessed 23-11-2014].

[4] Andy Orchard. *Dictionary of Norse Myth and Legend*. Cassell, 1997.

INSA Rennes
20 Avenue des Buttes de Coësmes
CS 70839
35708 Rennes Cedex 7

Tél. +33 (0) 2 23 23 82 00
Fax +33 (0) 2 23 23 83 96

www.insa-rennes.fr

INSA

UNIVERSITÉ EUROPÉENNE DE BRETAGNE

Commission des Titres d'Ingénieur

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE