KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Flip the virus: Modelling targeted attacks using FlipIt with propagation delay

Sophie Marien

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdspecialisatie Veilige software

Promotor:
Prof. dr. T. Holvoet

Assessoren:
Prof. dr. B. Jacobs
Dr. ir. A. Dries

Begeleiders:
Ir. Jonathan Merlevede,
Ir. Kristof Coninx

Academiejaar 2014 – 2015

Preface

Many people have helped in the realisation of this thesis. I would like to thank everybody who kept me busy the last year and made it possible for me to finish my thesis.

First of all, I would like to thank my promoter and my assistants for their support during the writing of this paper. Without their guidance this paper would not have been possible. They helped me figure out what to do and put a lot of time and effort in this work.
Special thanks to my parents for giving me the opportunity to study in this wonderful city and obtaining my master's degree in engineering.
I would also like to thank my boyfriend Jeroen, who supported me during the writing of this paper and helped me not to flip out.
Last but not least, I would like to thank all the people that read my paper for typo's, mistakes, and ease of understanding: my mother, my brother Hans, and my friends Matthias, Ska and Stijn.

A few other people earn a special mention: Revue-blokt for keeping me focused on my paper, DistriNet labo for letting me in and allowing me to study in a quite place and Antonio for distracting me and telling me stories about his travels. A special tribute to John Nash, a mathematician with a fundamental contribution to Game Theory, who died in a car accident during the writing of this thesis. His Nash Equilibrium is the main motivation behind the application of game theory to security problems in this paper.

Sophie Marien

Contents

Abstract

Recently, high profile targeted attacks such as the attack on Belgacom (a major Belgian Telecom), have demonstrated that even the most secure companies can still be compromised, and moreover that such attacks can go undetected for a while. This kind of attack is called an APT, Advanced Persistent Threat, and is designed to secretly penetrate a computer network, collect sensitive data and stay hidden for many years. Companies have every interest to mitigate the risks of an APT and the consequences that it can cause. Because of stealthiness, fighting against this kind of attack requires methods that go beyond the standard tools against malware.
A group of researchers at the RSA, van Dijk et al., proposed the game FlipIt (The game of "stealthy takeover") to model stealthy takeovers. It is a 2-players game composed of a single attacker, a single defender and a single shared resource. The players will compete to get control over the shared resource. Every move of the players will involve a cost and these moves happen in a stealthy way. The objective of the game for each player is to maximise the fraction of time being in control of the resource and to minimise the total move cost.
FlipIt does however not take into account that a move may not be instantaneous, but may have a certain delay. We adapt FlipIt such that we can use it to model the game of defending a company network that is attacked by an APT. The FlipIt formulas are adapted such as to take the delay for an APT propagation into account, which in our case will be a delay for the attacker. In this paper, we restrict ourselves to games where both the defender and the attacker play with a periodic strategy. The goal of this paper is to find out if modelling such situations with FlipIt with propagation delay allows us to draw interesting lessons about security measures against APTs.

Keywords: Game theory, Advanced Persistent Threats, cyber security, FlipIt, stealthy takeovers, propagation methods.

Samenvatting

In onze hedendaagse maatschappij valt security niet weg te denken. Door de technologische vooruitgang worden security-aanvallen veel geavanceerder en moeilijker te bestrijden. Zo zijn er onlangs gerichte security-aanvallen geweest op grote bedrijven, zoals bijvoorbeeld de aanval op Belgacom (een grote Belgische telecom). Deze aanvallen hebben aangetoond dat zelfs de meest veilige bedrijven nog steeds gecompromitteerd kunnen worden, en dat bovendien dergelijke aanvallen onopgemerkt kunnen blijven gedurende een hele tijd.

Een groot aantal bedrijven hebben databases die belangrijke informatie bevatten zoals bijvoorbeeld vertrouwelijke informatie over klanten. Het is belangrijk dat deze informatie binnen het bedrijf blijft. Het doel van computer en network security is om deze informatie te beschermen tegen bedreigingen.
Deze thesis focust zich op Advanced Persistent Threats (APT). Een APT is een niet-aflatende en gerichte cyber-aanval die ontworpen is om systemen en netwerken heimelijk binnen te dringen en dan voor een lange tijd onopgemerkt te blijven. Een manier om deze heimelijke aanvallen te analyseren is door deze te modelleren via speltheorie. Speltheorie krijgt meer en meer belangstelling in het veld van security om cyber security problemen te analyseren. Deze problemen zijn meestal gemodelleerd als een spel met twee spelers: een aanvaller en een verdediger.

Deze thesis bouwt verder op een security spel geïntroduceerd door van Dijk et al, “FlipIt” [24]. FlipIt is een speltheoretisch framework om scenario’s te modelleren die een heimelijk aspect hebben. Het is een 2-spelers spel bestaande uit een aanvaller, een verdediger en een gedeelde bron. De spelers proberen om controle te krijgen over de gedeelde bron en ze doen dit op een heimelijke manier. FlipIt houdt echter geen rekening met het feit dat een aanval niet onmiddellijk effect heeft, maar dat de feitelijke overname van de bron kan gebeuren met een zekere vertraging. Het kan bijvoorbeeld zijn dat een virus even tijd nodig heeft om een computer over te nemen. In deze thesis passen we het model van FlipIt zodanig aan dat het toepasbaar is voor heimelijke aanvallen die onderhevig zijn aan een vertraging.

FlipIt

Om te begrijpen hoe we het FlipIt spel kunnen aanpassen om APT propagatie in acht te nemen, is het belangrijk om vertrouwd te raken met de concepten van het

basisch FlipIt spel en de notaties.

FlipIt is een spel met twee spelers met een gedeelde bron die de spelers zo lang mogelijk willen beheren. De gedeelde bron kan een wachtwoord, een netwerk of een geheime sleutel zijn, afhankelijk van de situatie. In de rest van de thesis duiden we de twee spelers aan met onderschrift A voor de aanvaller en onderschrift D voor de verdediger.

Het spel begint op tijdstip $t = 0$ en blijft voor onbepaalde tijd doorgaan ($t \to \infty$). De tijd van het spel is continu. Om controle over de bron te krijgen kunnen de spelers i, met $i \in \{A, D\}$ de bron flippen. Elke flip impliceert een zekere kost k_i en deze kosten kunnen variëren voor elke speler. Beide spelers proberen om hun kosten te minimaliseren. Door een kost in te voeren, voorkomt men dat de spelers te vaak flippen.

De unieke eigenschap van FlipIt is dat elke flip op een heimelijke manier gebeurt. Dit betekent dat beide spelers geen weet hebben over wie de controle heeft over de bron. Zo zal de verdediger niet kunnen achterhalen of de bron al is gefipt door de aanvaller tot hij de bron zelf flipt. Het doel van elke speler is om zo lang mogelijk de controle te houden over de bron en tegelijkertijd de kost van de bewegingen (het flippen) te minimaliseren. Een beweging kan ook leiden tot een "verloren beweging", genoemd een flop. Als een speler flipt wanneer hij of zij al de controle heeft over de bron, dan verspilt deze speler een zet omdat het niet leidt tot een verandering van controle. Er gaat dus ook een kost verloren.

Figuur 1: Een FlipIt spel waarbij beide spelers periodiek spelen. Elke beweging of flip is aangegeven met een groene (donkergrijs) of oranje (lichtgrijze) cirkel. De aanvaller is weergegeven in het oranje en speelt met een periode van $\delta_A = 4$. De verdediger is weergegeven in het groen en speelt met een periode van $\delta_D = 3$. De groene en oranje rechthoeken geven aan welke speler in controle is van de bron.

Een tijdsafhankelijke variabele $C = C_i(t)$ duidt de toestand van de bron aan. $C_D(t) = 1$ als het spel onder controle is van de verdediger en 0 als het spel onder controle is van de aanvaller. Analoog zal $C_A(t) = 1$ zijn als het spel onder controle is van de aanvaller en 0 als het onder controle is van de verdediger. Aangezien er op elk tijdstip een en slechts een speler in controle is over de bron is dus $C_A(t) + C_D(t) = 1$. Het spel begint met de verdediger in controle: $C_D(0) = 1$.

De totale winst van de speler i is gelijk aan de totale hoeveelheid tijd waarin een

speler i in controle is van de bron vanaf het begin van het spel tot de huidige tijd t:

$$G_i(t) = \int_0^t C_i(x)dx. \tag{1}$$

De totale winst van de verdediger opgeteld bij de totale winst van de aanvaller telt op tot t:

$$G_D(t) + G_A(t) = t \tag{2}$$

De gemiddelde winst ratio van speler i is als volgt:

$$\gamma_i(t) = G_i(t)/t. \tag{3}$$

En daarmee geldt voor alle $t > 0$:

$$\gamma_D(t) + \gamma_A(t) = 1 \tag{4}$$

De spelers krijgen een benefit gelijk aan de hoeveelheid tijd dat ze in het bezit zijn van de bron min de kosten voor het maken van de bewegingen. $\beta_i(t)$ is de gemiddelde benefit ratio van een speler i tot aan tijd t:

$$\beta_i(t) = \gamma_i(t) - k_i\alpha_i. \tag{5}$$

waarin de kosten van een speler i worden aangegeven met k_i en waarin α_i het gemiddelde flipfrequentie definieert voor speler i met $n_i(t)$ het aantal bewegingen door speler i:

$$\alpha_i(t) = \frac{n_i(t)}{t} \tag{6}$$

De benefitratio is gelijk aan de fractie van de tijd waarin de bron in handen is van speler i, minus de kost van het flippen. Gedurende het spel wordt de asymptotische benefit ratio (of ook gewoon benefit) gedefinieerd als *lim inf* van de gemiddelde benefit omdat de tijd t toeneemt tot oneindig en de gemiddelde benefit niet altijd een limiet heeft.

$$\beta_i(t) = \lim_{t\to\infty} inf\beta_i(t) \tag{7}$$

Strategieën

Omdat de spelers op een heimelijke manier bewegen, zijn er verschillende soorten feedback die een speler kan krijgen tijdens het flippen. Dergelijke feedback verdeelt de strategiën in twee groepen: de niet-adaptieve strategieën en de adaptieve strategieën. Deze zijn beschreven in de tabel 1.

Indien een speler geen feedback krijgt, zal hij op dezelfde manier spelen tegen elke tegenstander. Deze strategie noemt men een niet-adaptieve strategie omdat de speelstrategie niet afhankelijk is van de bewegingen van de tegenstander. Een interessante subklasse van de niet-adaptieve strategieën is de *renewal* strategie waarbij

Categoriën	**Klassen**
Niet-adaptieve (NA)	Renewal
	- Periodieke
	- Exponentiële
	Algemeen niet-adaptieve
Adaptieve (AD)	Last move (LM)
	Full History (FH)

Tabel 1: Hiërarchie van de strategiën in FlipIt

de tijdsintervallen tussen twee opeenvolgende bewegingen worden gegenereerd door een vernieuwingsproces. Het vernieuwingsproces betekent dat de strategiën worden vernieuwd na elke beweging: de lengte van het interval tot aan de volgende beweging hangt alleen af van de huidige tijd en niet van de voorafgaande geschiedenis. Een voorbeeld van een dergelijke *renewal* strategie is de periodieke strategie waarbij het tijdsverloop tussen twee opeenvolgende bewegingen van de spelers bepaald is door een vast interval. Een exponentiële strategie is een *renewal* strategie waarbij het interval tussen twee opeenvolgende zetten exponentieel verdeeld is.

Als een speler feedback krijgt, kan hij zijn strategie aanpassen aan de informatie verkregen over de bewegingen van de tegenstander. Afhankelijk van de feedback kunnen twee subklassen van adaptieve strategieën worden geïdentificeerd. The *Last Move* (LM) strategiën vertegenwoordigen de klasse waarbij een speler de exacte tijd te weten komt van de laatste flip van de tegenstander. In de tweede klasse, genaamd *Full History* (FH), krijgt een speler de hele geschiedenis van de beweging van de tegenstander wanneer hij flipt.

Uit het onderzoek van het FlipIt framework zijn er een aantal interessante resultaten bekomen:

- periodieke spellen domineren de andere *renewal* strategiën. Dit betekent dat het altijd voordeliger is om een periodieke strategie te spelen tegen een tegenstander met een *renewal* strategie;
- periodiek spelen is nadelig tegen spelers die de *Last Move* adaptieve strategie gebruiken. De tegenspeler kan telkens te weten komen wanneer de volgende flip zal komen en juist erachter zelf flippen. De *Last Move* tegenspeler zal dus heel de tijd in controle zijn van de bron met verwaarloosbare onderbrekingen door de andere speler;
- als een speler speelt tegen een *Last Move* tegenspeler die een veel hogere kost heeft, dan heeft de speler twee opties. De eerste optie is om zo snel te spelen zodat de hoge kosten de tegenspeler dwingt om te stoppen met het spel. De tweede optie is om te spelen met een random strategie, zodat de tegenspeler niets kan leren uit de informatie betreffende de volgende beweging van de speler;

- de beste verdedigingsstrategie is om snel te spelen, om er zo voor te zorgen dat de tegenspeler afhaakt. Om snel te spelen, moet de speler er voor zorgen dat zijn bewegingskosten veel kleiner zijn dan de kosten van de tegenspeler.

FlipIt met Propagatie Vertraging

Het FlipIt spel bestaat uit een enkele bron. Om het security probleem voor te stellen van een APT die propageert doorheen een netwerk, definiëren we de bron als een computer netwerk met meerdere knooppunten. Een van de spelers, de verdediger, zal proberen om zijn netwerk te verdedigen. Hij zal dit doen door elk knooppunt van het netwerk te flippen. De andere speler, de aanvaller, zal proberen om alle knooppunten in het netwerk te infecteren. De aanvaller zal dit doen door een APT te droppen op een van de knooppunten in het netwerk. Deze APT zal zich dan verspreiden en andere knooppunten in het netwerk infecteren.

Na het droppen van een APT op een knooppunt in het netwerk duurt het een tijdje voor de malware het gehele netwerk heeft geïnfecteerd. Dus de veronderstelling dat de aanvaller de volledige controle heeft over het hele netwerk zodra een knooppunt besmet is, is niet realistisch. De aanvaller heeft slechts controle over heel het hele netwerk van zodra er een voldoende aantal knooppunten besmet zijn. De tijd die nodig is voor de APT om elk knooppunt te infecteren (of een voldoende aantal knooppunten) wordt aangeduid als een infectie vertragingsvariabele d. Om de waarde van d te berekenen, is het voldoende om het kortste pad te berekenen tussen het eerst geïnfecteerde knooppunt en het knooppunt dat het verst hiervan af staat. De variabele d kan ook dienen om de tijd aan te duiden die nodig is om een voldoende aantal knooppunten te infecteren.

Veronderstel dat een aanvaller aanvalt op moment t. Hij krijgt niet onmiddellijk de controle over het netwerk, maar pas na $t + d$. Als de verdediger flipt voordat de periode d verlopen is (ergens tussen t en $t+d$), dan zal de aanvaller nooit de volledige controle krijgen over het netwerk. Dit impliceert dat de wiskundige formules voor de *gain* en de *benefit* aangepast moeten worden aan het feit dat de aanvaller een deel van zijn winst verliest vanwege de vertraging d. De rest van deze thesis zal zich toeleggen op de formalisering van het FlipIt spel met behulp van de variabele d.

De formalisering begint bij het model van het niet-adaptieve continue basis FlipIt spel waarin spelers een periodieke strategie gebruiken met een willekeurige fase. De motivatie voor deze keuze is de veronderstelling dat in de praktijk bij de meeste bedrijven de verdedigingsstrategie om het netwerk te verdedigen periodiek is. Er is gekozen voor een periodieke aanvallersstrategie zodat we in staat zijn om de resultaten te vergelijken met de periodieke strategie van het spel FlipIt in [24]. Bovendien weten we uit de resultaten van het basisch FlipIt spel dat de periodieke strategie de beste strategie is binnen de categorie van de renewal strategiën. De adaptieve strategiën zijn minder relevant gezien het heimelijk karakter van de verdediger en de aanvaller.

De verdediger zal geen informatie hebben over het tijdstip van de aanval en andersom.

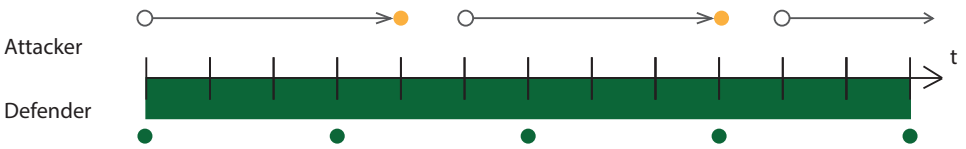
We delen de formalisatie op in drie gevallen. Gelijkaardig als in FlipIt [24], is er het geval waarbij de verdediger minstens zo snel speelt als de aanvaller en in het andere geval waarbij de aanvaller minstens zo snel speelt als de verdediger. Voor beide gevallen presenteren we eerst de *gain* formule van een FlipIt spel zonder vertraging, en daarna introduceren we de vertraging. Het derde geval is een speciaal geval waarbij de periode van de verdediger trager of gelijk is aan de vertraging.

Formalisering van de benefitformule met een vertraging

Een periodische strategie is een niet-adaptieve strategie waarin het tijdsinterval tussen twee opeenvolgende bewegingen een constante is, weergegeven als δ. Bovendien heeft het een random fase die uniform en random gekozen is in het eerste interval tijdens de eerste flip $[0, \delta]$. De gemiddelde snelheid van de bewegingen van een speler is uitgedrukt als volgt: $\alpha_i = \frac{1}{\delta_i}$.

Case 0: $\delta_D \leq d$ (De verdediger speelt met een periode trager of gelijk aan de vertraging

Wanneer de vertraging groter of gelijk is aan de periode van de verdediger , dan zal de aanvaller nooit controle krijgen over de bron. The verdediger zal altijd flippen voordat de vertraging verlopen is. Figuur 2 stelt de situatie voor.



Figuur 2: FlipIt met een vertraging die groter is dan de snelheid van de verdediger: $d \geq \delta_D$

In dit geval is de benefit voor de verdediger en de benefit voor de aanvaller gelijk aan het volgende:

$$\beta_D(\delta_D, \delta_A) = 1 - \frac{k_D}{\delta_D} \tag{8}$$

$$\beta_A(\delta_D, \delta_A) = -\frac{k_A}{\delta_A} \tag{9}$$

In de rest van de cases zal δ_D altijd groter zijn dan de vertraging d.

Case 1: $\delta_D \leq \delta_A$ (De verdediger speelt minstens zo snel als de aanvaller.)

Stel $r = \frac{\delta_D}{\delta_A}$. The intervallen tussen twee opeenvolgende bewegingen hebben lengte δ_D. Beschouw een interval van de verdediger. De waarschijnlijkheids-verdeling over

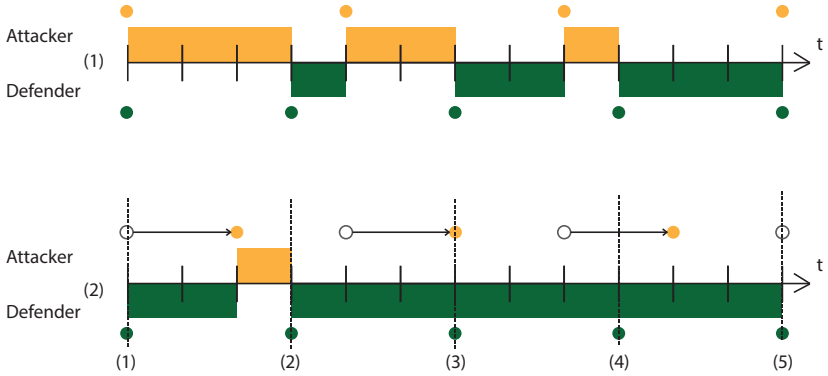
de fase van de verdediger dat de aanvaller flipt tijdens dit interval is $\frac{\delta_D}{\delta_A} = r$. Gegeven dat de aanvaller flipt in dit interval, zal hij exact een keer flippen in dit interval (aangezien $\delta_D \leq \delta_A$) en zijn bewegingen uniform en random verdeeld zijn.

De verwachte periode dat de aanvaller controle heeft in dit interval is $r/2$, zonder de vertraging door een APT in acht te nemen. Daarom kan de benefit voor de aanvaller, zonder de vertraging als volgt uitgeschreven worden:

$$\beta_A(\alpha_D, \alpha_A) = \frac{r}{2} - k_A\alpha_A = \frac{\delta_D}{2\delta_A} - k_A\alpha_A \tag{10}$$

Bijgevolg is de benefit voor de verdediger:

$$\beta_D(\alpha_D, \alpha_A) = 1 - \frac{r}{2} - k_D\alpha_D = 1 - \frac{\delta_D}{2\delta_A} - k_D\alpha_D \tag{11}$$



Figuur 3: Het eerste spel is een spel zonder APT propagatie. Het tweede spel is een spel met APT propagatie en $d = 1$. De vertraging wordt aangeduid via een pijl.

Echter, omwille van de vertraging die nodig is door de APT propagatie, is de maximale tijd dat de aanvaller in controle kan zijn verminderd van δ_D naar $\delta_D - d$. Zie figuur 3.5. Er is een kans r dat de aanvaller zal flippen in het interval van de verdediger. Door de vertraging zal de *gain* deze keer niet de helft zijn van het interval. De aanvaller moet vroeg genoeg spelen om controle te krijgen over de bron. Vroeg genoeg betekent dat de aanvaller moet spelen tijdens de periode van $\delta_D - d$ tijdens het interval van de verdediger. De kans dat de aanvaller vroeg genoeg zal spelen is $\frac{\delta_D - d}{\delta_D}$ en dit geeft de aanvaller een gemiddelde *gain* van $\frac{\delta_D - d}{2}$. Als de aanvaller flipt na de periode van $\delta_D - d$, dan zal zijn *gain* gelijk zijn aan nul. De kans dat de aanvaller te laat flipt is gelijk aan $\frac{d}{\delta_D}$. De gemiddelde *gain* ratio van de aanvaller kan

dus als volgt uitgedrukt worden, als ze kijken naar het interval van de verdediger:

$$\gamma_A(\delta_D, \delta_A) = \frac{1}{\delta_D}[\frac{\delta_D}{\delta_A} \cdot [\frac{\delta_D - d}{\delta_D} \cdot \frac{\delta_D - d}{2} + \frac{d}{\delta_D} \cdot 0]]$$

Om de benefit te berekenen, trekken we de bewegingskosten af van de gemiddelde *gain*.

$$\beta_A(\delta_D, \delta_A) = \frac{(\delta_D - d)^2}{2 \cdot \delta_D \delta_A} - k_A \alpha_A \tag{12}$$

De benefit van de verdediger wordt dan als volgt uitgedrukt:

$$\beta_D(\delta_D, \delta_A) = 1 - \frac{(\delta_D - d)^2}{2 \cdot \delta_D \delta_A} - k_D \alpha_D \tag{13}$$

We kunnen zien dat wanneer $d = 0$, de formule terug gelijk is aan de formule in het originele FlipIt spel [24].

Case 2: $\delta_A \leq \delta_D$ (De aanvaller speelt minstens zo snel als de verdediger)

Stel $r = \frac{\delta_D}{\delta_A}$. De lengte van de intervallen tussen twee opeenvolgende bewegingen van de aanvaller zijn gelijk aan δ_A. Beschouw een gegeven interval van de aanvaller. De waarschijnlijkheids-verdeling over de fase van de aanvaller dat de verdediger flipt tijdens dit interval is $\frac{\delta_A}{\delta_D} = (1/r)$. Gegeven dat de verdediger flipt in dit interval, zal hij exact een keer flippen in dit interval (aangezien $\delta_A \leq \delta_D$) en zijn bewegingen zijn uniform en random verdeeld.

Er volgt uit een gelijkaardige analyse als in case 1 voor een FlipIt spel zonder virus propagatie met de volgende benefit formules:

$$\beta_D(\alpha_D, \alpha_A) = \frac{1}{2r} - k_D \alpha_D = \frac{\delta_A}{2\delta_D} - k_D \alpha_D \tag{14}$$

$$\beta_A(\alpha_D, \alpha_A) = 1 - \frac{1}{2r} - k_A \alpha_A = 1 - \frac{\delta_A}{2\delta_D} - k_A \alpha_A \tag{15}$$

Voor de case met een virus propagatie beschouwen we twee onderverdelingen: case 2.a en case 2.b, afhankelijk of de vertraging langer of korter duurt dan het verschil tussen de periode van de aanvaller en de verdediger.

Case 2.a: $d + \delta_A \leq \delta_D$

Beschouw een tijdsperiode van $\delta_A + d$, die het interval van de aanvaller voorstelt gevolgt door de periode van de vertraging in het volgende interval. De verdediger zal nooit twee keer flippen in dit interval omdat $\delta_A + d \leq \delta_D$. De verdediger zal

flippen tijdens het interval met een kans van $\frac{\delta_A}{\delta_D}$. Wanneer de verdediger flipt, zal de verdediger tot op het einde van het interval in controle zijn van de bron. In het volgende interval zal de aanvaller terug moeten flippen om terug controle te krijgen. Dit betekent dat tijdens de vertraging in het volgende interval, de verdediger de controle zal hebben, zie figuur 3.7 cases (1) and (2). De totale tijd dat de verdediger controle heeft over de bron is vanaf het moment dat de verdediger flipt tot het einde van het interval plus de periode van de vertraging in het volgende interval, namelijk d. Omdat $d + \delta_A \leq \delta_D$ zal de volgende flip van de verdediger nooit gebeuren tijdens de vertraging. De hele vertraging kan worden beschouwd als een extra benefit van een flip in het vorige interval. Elke keer dat de verdediger flipt, zal hij een gemiddelde *gain* hebben van $\frac{\delta_A}{2}$ in het interval waarin hij flipt en een extra *gain* van d in het volgende interval. Dit resulteert in een totaal gemiddelde *gain* per interval van $\frac{(d + \frac{\delta_A}{2})}{\delta_A}$.

De totale *gain* rate van de verdediger is dan de kans dat de verdediger zal flippen gedurende het interval van de aanvaller vermenigvuldigd met de totaal gemiddelde *gain* per interval:

$$\gamma_D(\alpha_D, \alpha_A) = \frac{\delta_A}{\delta_D} \cdot \frac{(d + \frac{\delta_A}{2})}{\delta_A} \tag{16}$$

$$\gamma_D(\alpha_D, \alpha_A) = \frac{\delta_A}{2\delta_D} + \frac{d}{\delta_D} \tag{17}$$

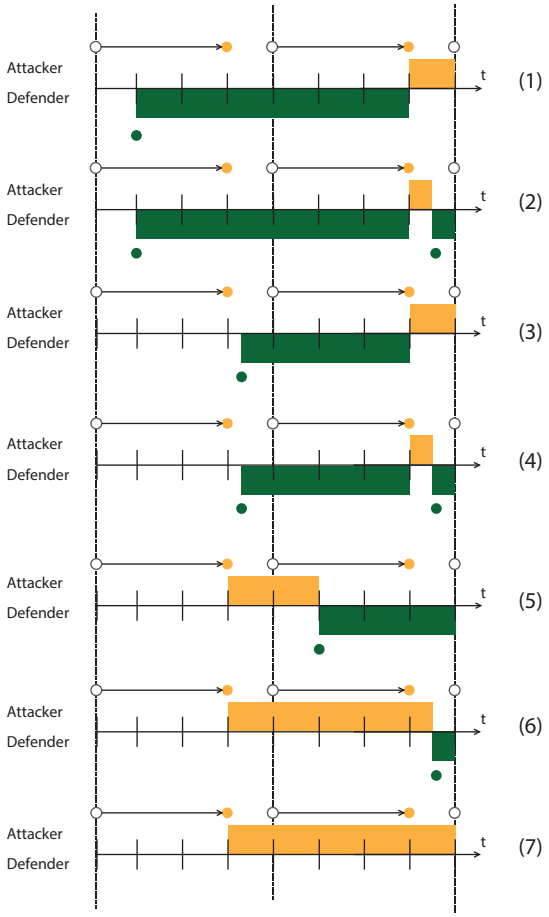
Dit resulteert in de volgende benefit formule:

$$\beta_D(\alpha_D, \alpha_A) = \frac{\delta_A}{2\delta_D} + \frac{d}{\delta_D} - k_D\alpha_D \tag{18}$$

De benefit van de aanvaller is dan als volgt:

$$\beta_A(\alpha_D, \alpha_A) = 1 - \frac{\delta_A}{2\delta_D} - \frac{d}{\delta_D} - k_A\alpha_A \tag{19}$$

Het is cruciaal dat δ_D op zijn minst zo groot is als $d + \delta_A$. Zo niet betekent dit dat de aanvaller kan bewegingen tijdens de vertraging in het interval dat volgt op het interval waar de verdediger al geflipt heeft. Stel dat dit wel gebeurt dan kan er een overlap zijn tussen de gemiddelde *gain* van $\frac{\delta_A}{2}$ en de vertraging. De bovenstaande formule bevat dan een overschatting van de *gain* voor de verdediger gelijk aan de overlap tijdens de vertraging die dubbel geteld wordt.



Figuur 4: Alle verschillende situaties voor de verdediger en de aanvaller in case 2.a waar $d + \delta_A \leq \delta_D$ Van case (1) tot case (4) zal de verdediger de controle hebben gedurende een perdiode van d in het interval dat volgt op het interval waarin de verdediger geflipt heeft.

Case 2.b: $d + \delta_A \geq \delta_D$

Om de formule te bekomen voor de case waarbij de vertraging te groot is, moeten we de overlap met de *gain* berekenen en deze aftrekken. Aangezien $\delta_D \geq \delta_A$, als de verdediger onmiddellijk speelt nadat de aanvaller geflipt heeft, de verdediger nooit in het voorgaande interval geflipt hebben. In dit geval is er geen overlap. Het probleem van de overlap doet zich alleen voor als de verdediger te laat in het interval flipt. Dit betekent dat alleen het laatste deel van de vertraging kans heeft tot overlap. Hoe groter het verschil tussen de periode van de aanvaller en van de verdediger, hoe kleiner de kans tot overlap. Concreet gezien, alleen het laatste deel met lengte $d-(\delta_D-\delta_A)$ is onderworpen aan overlap. Hieruit volgt dat de kans tot overlap gelijk is aan $\dfrac{d-(\delta_D-\delta_A)}{\delta_D}$ en dat de *gain* de helft zal zijn van dit interval: $\dfrac{d-(\delta_D-\delta_A)}{2}$.
De *gain* rate die voor overlap zorgt en dus afgetrokken moet worden is:

$$\frac{1}{\delta_A} \cdot \frac{d - (\delta_D - \delta_A)}{\delta_D} \cdot \frac{d - (\delta_D - \delta_A)}{2} \tag{20}$$

De totale *gain* rate van de verdediger bekomt men dus door de overlap af te trekken van de *gain* rate die bekomen is in case a:

$$\gamma_D(\alpha_D, \alpha_A) = \frac{\delta_A}{\delta_D} \cdot \frac{(d + \frac{\delta_A}{2})}{\delta_A} - \frac{(d - (\delta_D - \delta_A))^2}{2\delta_D\delta_A} \tag{21}$$

$$\gamma_D(\alpha_D, \alpha_A) = \frac{\delta_A}{2\delta_D} + \frac{d}{\delta_D} - \frac{(d - (\delta_D - \delta_A))^2}{2\delta_D\delta_A} \tag{22}$$

Hieruit volgt de volgende benefit formule:

$$\beta_D(\alpha_D, \alpha_A) = \frac{\delta_A}{2\delta_D} + \frac{d}{\delta_D} - k_D\alpha_D - \frac{(d - (\delta_D - \delta_A))^2}{2\delta_D\delta_A} \tag{23}$$

De benefit voor de aanvaller is de volgende formule:

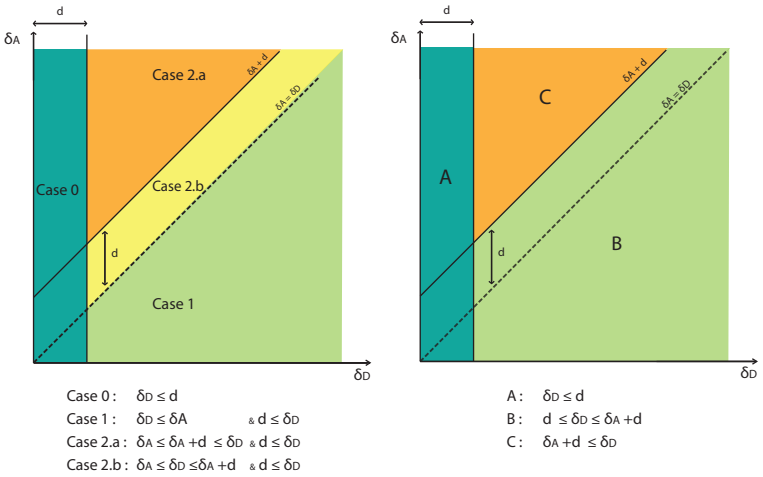
$$\beta_A(\alpha_D, \alpha_A) = 1 - \frac{\delta_A}{2\delta_D} - \frac{d}{\delta_D} - k_A\alpha_A + \frac{(d - (\delta_D - \delta_A))^2}{2\delta_D\delta_A} \tag{24}$$

Samenvatting

We berekenden de benefit voor beide spelers voor de volgende drie gevallen: wanneer δ_D kleiner is dan d, wanneer δ_D kleiner is dan δ_A en wanneer δ_D groter is dan δ_A. De benefit formules voor case 1 en case 2.b leiden tot hetzelfde resultaat. Deze twee cases kunnen worden samengebracht door het aanpassen van de randvoorwaarden. Op figuur 5 toont de linkse voorstelling de cases die samengebracht kunnen worden. Case 1 en case 2.b liggen naast elkaar. Het resultaat van het samenbrengen is visueel te zien op de rechtse voorstelling in figuur 5.

De benefit functies werden als piecewise functies voor elke case uitgerekend:
De benefit formule voor de verdediger is als volgt:

$$\beta_D(\delta_D, \delta_A) = \begin{cases} 1 - \frac{k_D}{\delta_D}, & \delta_D \leq d \\ 1 - \frac{(\delta_D - d)^2}{2\delta_D\delta_A} - \frac{k_D}{\delta_D}, & d \leq \delta_D \leq d + \delta_A \\ \frac{\delta_A}{2\delta_D} + \frac{d}{\delta_D} - \frac{k_D}{\delta_D} & \delta_D \geq d + \delta_A \end{cases}$$



Figuur 5: De eerste figuur is de voorstelling van alle cases waarvoor de benefit berekend is. De tweede figuur is de voorstelling van de nieuwe randvoorwaarden door het samenbrengen van case 1 en case 2.b.

De benefit formule voor de aanvaller is als volgt:

$$\beta_A(\delta_D, \delta_A) = \begin{cases} -\dfrac{k_A}{\delta_A}, & \delta_D \leq d \\ \dfrac{(\delta_D - d)^2}{2\delta_D \delta_A} - \dfrac{k_A}{\delta_A}, & d \leq \delta_D \leq d + \delta_A \\ 1 - \dfrac{\delta_A}{2\delta_D} - \dfrac{d}{\delta_D} - \dfrac{k_A}{\delta_A} & \delta_D \geq d + \delta_A \end{cases}$$

De piecewise functies zijn ook functies die continu zijn. Door de randgevallen in te vullen, (e.g $\delta_D = d$ voor case A en case B), bekomt ben dezelfde formule.

Optimale Strategiën voor de Aanvaller en de Verdediger

Na het bepalen van de benefit functies zijn we geïnteresseerd in de optimale strategiën van de aanvaller en de verdediger. Vanuit deze strategiën kunnen later de Nash evenwichten bepaald worden. De berekening van de optimale strategiën, gegeven een bepaalde periode van de tegenpartij zijn als volgt:

De functie $opt_D(\delta_A)$:

$$opt_D(\delta_A) = \begin{cases} [0,d] & \delta_A = 0 \ \ \& \ \ k_D = 0 \\ [0,d] & k_D = 0 \\ d & \delta_A = 0 \ \ \& k_D \leq d \\ \infty, & \delta_A < 2(k_D - d) \\ \left[\sqrt{2k_D\delta_A + d^2}, \infty\right[, & \delta_A = 2(k_D - d) \\ \sqrt{2k_D\delta_A + d^2}, & \delta_A > 2(k_D - d) \end{cases}$$

De functie $opt_A(\delta_D)$:

$$opt_A(\delta_D) = \begin{cases} \infty & \delta_D = 0 \\ [0,\infty] & \delta_D \leq d \ \ \& \ \ k_A = 0 \\ [0,\delta_D - d] & \delta_D > d \ \ \& \ \ k_A = 0 \\ \infty & \delta_D < d + k_A + \sqrt{2dk_A + k_A^2} \\ [\delta_D - d, \infty], & \delta_D = d + k_A + \sqrt{2dk_A + k_A^2} \\ \delta_D - d, & \delta_D > d + k_A + \sqrt{2dk_A + k_A^2} \end{cases}$$

Conclusie

Deze thesis presenteert een aanpassing aan het originele FlipIt spel [24] om de propagatie van een virus mee in rekening te brengen. We onderscheiden drie gevallen: het geval waarbij de verdediger trager of even snel speelt als de vertraging, het geval waarbij de verdediger trager of even snel speelt als de aanvaller en het geval waarbij de verdediger sneller of even snel speelt als de aanvaller. In het geval waarbij de verdediger sneller speelt als de aanvaller, zal de aanvaller telkens benefit verliezen door de vertraging. In het geval waarbij de aanvaller sneller speelt dan de verdediger, zal de verdediger telkens wanneer hij speelt, extra benefit krijgen door de vertraging van de aanvaller. Verder onderzoekzal moeten aantonen wat de exacte impact van de vertraging van de aanvaller heeft op de Nash evenwichten en het bepalen van de optimale strategie voor de verdediger en de aanvaller.

Het bepalen van de optimum functies voor de verdediger en de aanvaller van het FlipIt spel met de vertraging leidt tot interessante resultaten:

- Wanneer de verdediger sneller speelt dan de vertraging zal de aanvaller niet spelen. Zijn benefit in dit geval zal altijd negatief zijn. Een uitzondering hierop is als zijn kost nul is, dan is de benefit van de aanvaller nul en maakt het niet uit hoe snel hij speelt.
- Als de verdediger kan spelen met een kost gelijk aan nul, dan zal de aanvaller niet spelen. Doordat de kost nul is, kan de verdediger zo snel spelen als hij kan. Ook al probeert de aanvaller even snel te spelen als de verdediger, de vertraging zal altijd nadelig zijn voor de aanvaller.

- Vanaf een bepaalde snelheid van de aanvaller zal de verdediger niet meer spelen. Hetzelfde geldt voor een specifieke waarde van de verdediger, waarbij de aanvaller zal kiezen om niet meer te spelen.

APTs zijn steeds meer en meer bijzonder gesofisticeerde kwaadaardige stukken code. Er zijn APTs bekend die militaire disk-wiping en opnieuw formatteren overleven. Na het formatteren en het opnieuw installeren van het besturingssysteem kunnen deze APTs nog in staat zijn om gevoelige informatie door te sturen (Equation group [22]). Dit betekent dat zelfs als de verdediger weet dat een APT het netwerk geïnfecteerd heeft, de werkelijke praktische "flip" dient te bestaan en gekend moet zijn door de verdediger.
De meest efficiënte manier om een netwerk systeem te beveiligen is de zwakste link ontdekken. Dit zijn meestal de werknemers. Een effectieve manier om dit probleem tegen te gaan is om het bewustzijn over de risico's van infecties bij de medewerkers te verhogen. Zelfs in aanwezigheid van goede security protection layers, spamfilters en firewalls kan het openen van een phishing mail of gebruik van een onbekende USB stick door een van werknemers genoeg zijn om het netwerk te infecteren. Een systeem kan echter nooit 100 % waterdicht zijn. Indien een infectie heeft plaatsgevonden en de verdediger een tegenmaatregel kent, dan kan het Flipit spel helpen om het meest effectieve tempo te bepalen waarin tegenmaatregelen moeten worden doorgebracht.

List of Figures and Tables

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

RSA	Rivest-Shamir-Adleman
DDoS	Distributed Denial of Service
APT	Advanced Persistent Threat
CIA	Confidentiality, Integrity and Availability
USB	Universal Serial Bus
NA	Non-Adaptive
AD	Adaptive
LM	Last Move
FH	Full History
VM	Virtual Machine
IP	Internet Protocol
DNS	Domain Name System
BGP	Border Gateway Protocol
SEM	Simple Epidemic Model
SI	Susceptible - Infected
SIR	Susceptible - Infected - Removed
SIS	Susceptible - Infected - Susceptible
AAWP	Analytic Active Worm Propagation

Symbols

i	$i \in \{D, A\}$, defines the player. D is the defender and A is the attacker.
δ_i	The length of the interval between two consecutive moves of player i.
α_i	The average flip rate of player i, given by $\alpha_i = 1/\delta_i$.
k_i	The cost of player i's moves.
d	The delay caused by the propagation of a threat.
$G_i(t)$	The total gain of player i denotes the amount of time player i is in control over the resource up to time t.
γ_i	The average gain rate of player i defined as $G_i(t)/t$.
β_i	The average benefit rate up to time t defined as $\beta_i = \gamma_i - k_i\alpha_i$.
opt_i	The optimum function for player i.
$n_i(t)$	The amount of moves made by player i up to time t.

Chapter 1

Introduction

1.1 Introduction

In this era where digitalization becomes prominent in every aspect of our lives, where technology is growing fast and where businesses are always under attack, security becomes an issue of increasing complexity. Security is needed to protect websites, servers, applications, data, operating systems and other assets that need protection in a computer network. Without security, there is no protection to keep somebody out of a system. It is the same as leaving the door of your house wide open for everyone to come in.

Why is it so important to keep a system secure? Many businesses store confidential information, which can be lost through data leakage and can possibly be abused by competitors. Also, disruption caused by distributed denial of service (DDoS) attacks, may result in businesses failing to meet their service-level agreements. Ultimately, computer and network security helps protecting a business against various kind of threats.

A particular kind of threat is an Advanced Persistent Threat (APT). An APT is a multi-faceted, continuous and targeted cyber attack that is designed to penetrate a network or a system in a stealthy way and can stay undetected for a long period of time. It is different and more severe than a conventional threat. A conventional threat will not attack any particular target. An APT is persistent and will keep on trying to attack its victim. It operates silently and stealthily, to prevent detection. This makes it so hard to protect a network or a system against an APT.
There are a number of key strategies an organisation can apply to defend itself against APTs: awareness, whitelisting, system administration, network segregation, dynamic content checking and patch management. Nevertheless the combination of all these elements benefits from being complemented by other defence strategies to protect oneself against stealthy takeovers. One possible way to study the impact of stealthy takeovers and to determine practical recommendations for defenders is through game theory.

Game theory is gaining increasing interest as an effective technique to model and study cyber security problems. It is common to model cyber security problems as a game with two players, an attacker and a defender. There are, however, games that have more players e.g. when a third party is involved [3]. This paper focusses on a game with two players. The actions available to the attacker and the defender correspond respectively to the attacks on the system and the defensive measures that protect the system.

Many security games that bridge the gap between game theory and cyber security have already been investigated, so finding a new game can be challenging. This paper builds on a relatively new paper where the assumption of stealthiness is fairly unique, giving some interesting results.
The paper is from researchers at RSA, van Dijk et al., who presented a game-theoretic framework to model computer security scenarios called "FlipIt" [24]. They study the specific scenario where a system or network is repeatedly taken over completely by an attacker. This take-over is not immediately detected by the defender. It is a two-player game where the attacker and the defender are competing to get control over a shared resource. Neither player knows who is currently in control of the resource until they move. In FlipIt every move involves a cost and gives the player immediate control over the resource. The attacker will try to maximise the time that he controls the network, while the defender will try to maximise the time that the network is free of malware.
But what if the attacker moves and it takes some time before the attacker gets full control over the resource? FlipIt does not take into account that a move may not be instantaneous, but has a certain delay. Consider for example a network with different nodes (laptops, datacenters) as a resource. The attacker drops a virus on one of the nodes and waits until this virus infects the whole network. The attacker will only be in control of the resource when a sufficiently large amount of nodes of the network are infected. In this paper we present an adaptation of FlipIt to model a game where the moves of the attacker are not instantaneous. The formalization for this game starts from the model of non-adaptive continuous basic FlipIt game where players use a periodic strategy with a random phase.

Research questions

This paper adapts the model presented in [24] so as to take the delay for virus propagation into account. This leads us to the following research questions:

- How can we incorporate the notion of delay in the game-theoretical analysis of the FlipIt game for a periodic strategy?

- Does the resulting model allow an optimal defence strategy against an attacker?

Contributions and results

The following contributions are made in this paper:

- We propose an addition to the basic FlipIt model to model a scenario where the moves by the attacker will not be instantaneous. We extend the FlipIt game to a game wherein the attacker flips with a delay. The attacker only compromises the system if sufficient nodes in the network are infected.

- The periodic case of FlipIt is modelled with a delay, resulting in adapted optimum functions for the defender and the attacker. These adapted optimum functions yield a number of interesting cases.

- While the exact value of the delay for modelling the FlipIt game is not required, this paper complements the results of the FlipIt game with an overview of different propagation techniques of worms and models. The models are used to calculate the propagation delay (depending on the network layout). The paper also presents a method to calculate the speed of the propagation of a worm in a propagation independent of the topology of the network.

- The Page Rank matrix is introduced as a proof of concept to calculate the importance of every node in the network.

While it may seem trivial to extend the basic FlipIt model with a propagation delay, its mathematical treatment is not. In the paper of Laszka et al. [7] it seems that even a small extension adds to the already significant mathematical complexity. While the FlipIt game is quite symmetric, the mathematical complexity rises substantially by adding a propagation delay on the side of the attacker. Besides yielding more complex functions, the symmetry is also largely lost.

Overview of the thesis

The organisation of this paper is as follows. An introduction to cyber security and game theory is given in chapter 2. The chapter allows the reader to become familiar with the kind of threats that are in the scope of this work and the game theoretic concepts that will be further used in the paper. In the same chapter the FlipIt framework is summarized with its most important conclusions. The chapter concludes with an overview of the related work on FlipIt and further clarifies the contributions of this paper compared to existing work. Chapter 3 first introduces the adaptations made to the original FlipIt game to model a FlipIt game with a virus propagation delay. Subsequently formulas are derived to model a FlipIt game with a propagation delay for the specific case where players play a periodic strategy with a random phase.
In Chapter 4 the formulas are further analysed to determine optimal strategies for the defender and the attacker. In order to provide a clear perspective on delays, chapter 5 gives an overview of the various methods of propagation and worm propagation models. It presents a method to calculate the speed of the propagation of a worm

in a network independent of the topology of the network. It also explains how the PageRank algorithm may be used to calculate the sensitivity to infection of every node in the network. Finally chapter 6 discusses the main results and provides directions for further research.

Chapter 2

General Context

This chapter provides the reader with an introduction to the general context of the work presented in this paper. First, section 2.1 introduces the reader into the basic concepts of cyber security and the kind of cyber security threats that are in scope of this work. Section 2.2 then introduces the reader into the main principles of game theory. Subsequently, section 2.3 introduces the FlipIt game: this game will be used to model cyber security attacks of a periodic nature, including a delay. Finally, section 2.4 gives an overview of the related work, and how the research presented in this paper compares to existing results.

2.1 What Is Security?

Before the digitalization of documents, information was kept on paper and the security of this information was ensured by administrative and physical means. For example, you needed a key to access documents stored in a room full of cabinets where the files were kept. In today's digital era more and more information is kept in a digital format, stored on a computer. As digitalization progressed, the need for ensuring the security of digital information arose and automated tools were developed to protect files stored on a computer.

Information security is the generic term for protection of data stored on a computer controlled device such as computers and smartphones, as well as public and private computer networks, including the entire Internet. Security is a general term that encompasses several dimensions. More specifically, information security has three fundamental key objectives:

1. *Confidentiality*: assuring that the confidentiality of private data is not disclosed or made available to users that do not have proper authorization.

2. *Integrity:* assuring that data cannot be altered by an unauthorized individual.

3. *Availability:* assuring that data is always accessible and that the service is not denied to authorized individuals.

These key attributes are also known as the CIA triad. They are the fundamental security objectives for securing data, information and computing services.

Information security can be divided into two main subcategories. One of them is cyber security, also known as computer security. This is the process of applying security tools to ensure confidentiality, integrity, and availability of data. It is an attempt to protect websites, servers, data, applications, operating systems and all assets that need protection in a computer system. Some of these tools may include detection, identification or removal tools. A detection tool will determine if an infection has taken place and will trace the threat. An identification tool will try to identify the threat to be able to know how to remove it. A removal tool will remove the threat from the system (once it has been identified) so that it cannot spread any further.
The other category is network or internet security. This subcategory of information security protects data during transmission. While cyber security and network security address different aspects, they partially overlap as well. For example, a virus can be physically dropped on a computer network using a USB stick, but it can also arrive over the Internet. Either way internal computer security measurements have to be taken to recover from the virus. In this paper we focus on scenarios where a network has to be defended against attacks to ensure confidentiality, integrity and availability of data. The work presented in this paper therefore belongs to the domain of cyber security.

Threats to computer systems

The possible threats can take many forms, the most common being: spam, malware, spoofing, phishing and DDoS attacks. In the context of cyber security, the terms 'threat' and 'attack' are often used interchangeably, referring to more or less the same thing. The meaning of these two terms, however, differ slightly from one another. The former refers to anything that can breach security and cause possible harm. It is a possible danger that can exploit a vulnerability. The latter is an assault on computer security that originates from a threat. It is an intelligent act that deliberately tries to breach security through vulnerabilities.

The most noteworthy and biggest group of threats to computer systems is malware. This is a piece of malicious software that is designed to penetrate unprotected or vulnerable systems or computers, with the intent to retrieve sensitive information, destroy data, or compromise the confidentiality, integrity or availability of the data or applications of the victim. A security report of Kaspersky in 2014 [5] reveals that 61% of the attacks on companies are caused by malware. For this reason this section will examine the categories of malware threats. Different types of malware exist: viruses, worms, flooders, rootkits, bots, spyware, adware and many more. This broad range of different types can be classified into two main categories. The first one based on the propagation method that is used and the second one based on the payload or

the variety of actions that the malware performs [20]. Propagation methods include viruses, worms and trojans. Payload includes e.g. flooders, rootkits, bots, spyware and adware. This paper focuses on the category of propagation methods and not on the actions that the malware performs. A brief explanation of the three main propagation methods of malware is given below.

Virus: This is a malicious piece of code that replicates itself and tries to spread in order to infect other systems or files. A typical virus will attach itself to a program, or executable content on a computer. The *I love you* virus is an example of a virus where the virus attached itself as an executable to a mail. It propagated by using the mailing systems. When a victim opened an email with the *I love you* virus in the annex, the virus spread itself by sending a mail to everyone in the victim's contact list.

Using this method, a virus can multiply rapidly, possibly even causing a business network to shut down by the heavy traffic. A virus needs human interaction to spread. In the example above: if no one were to open the mail, the virus would not be able to spread itself and infect other systems.

Worm: A worm is a virus that can spread without human interaction. It is a computer program that replicates itself in order to spread to other hosts on a network. Copies of the worm can be forwarded via a computer network without an intermediary. The worm will use vulnerabilities of the system to infect other computers. The *Stuxnetworm* is a very prominent example of a worm. Initially this worm was spread via infected USB sticks and from there on it could spread itself to other hosts on the network through the Internet, without any further human intervention. The purpose of the *Stuxnetworm* was to harm the centrifuges in nuclear reactors; many reactors have been infected.

Trojan: This is a malicious program that disguises itself as something normal and useful, so that users won't be suspicious of installing it, but it has a malicious function hidden inside that can circumvent security measures and cause harm. A notable trojan horse is *Koobface*, that targeted users of Facebook, Skype, Yahoo, Gmail and AOL mail. To spread itself the trojan sent a mail or friend request with a message that directed the recipients to a third party website. This site would then convince the recipient into downloading an update of Adobe Flash Player. Once downloaded and executed, *Koobface* could infect the host.

As proposed by Kasperksy [6], threats caused by the malware described above can be divided into three main categories: known threats (70%), unknown threats (29%) and advanced threats (1%).

The known threats are the easiest to defend oneself against. Standard malware protection tools like firewalls and virus scanners can keep these kind of malware out of the system. Installing protection against unknown threats is also relatively

easy, but this requires tools that go beyond the standard methods, e.g. dynamic whitelisting. The remaining 1% are the advanced threats, also known as APTs. They are the most difficult to deal with.

Advanced threats to computer systems

An Advanced Persistent Threat (referred to for the remainder of this work as APT) is a persistent targeted attack that tries to penetrate a network to cause harm while staying unseen for a long period of time. The motive of an APT is mostly cyber espionage, stealing sensitive data, sabotage or other kinds of ideological attacks. APTs are 'advanced' because these attacks are well funded and because the attacker (usually) needs a great amount of expertise to successfully penetrate a network. Not all APTs are technically advanced though. The attacker can also try to exploit known vulnerabilities, in the hope that his target has not yet secured itself against them.
'Persistent' refers to the fact that the attacker isn't trying to gain immediate results, and the attacker won't stop trying after one failed attempt. The attack can be spread over several years, taking multiple steps.
An APT can be a mix of different types of malware and may use various propagation methods.

According to a security survey of Kaspersky [5] the damage of one successful targeted attack against a large company can exceed 2.5 million dollars. As such, companies need a defence mechanism to defend themselves against APTs. As previously stated, simple detection and identification tools are insufficient to protect oneself against APTs, and a removal tool will only work if the threat has been identified. To mitigate these kind of attacks another security countermeasure is needed.
This paper proposes an appropriate game-theoretical modelling of defending a network against APTs and will analyse it in order to draw the necessary strategic conclusions to mitigate these kind of attacks.

2.2 A Brief Introduction in Game Theory

Game theory is a mathematical study to analyse interactions between independent and self-interested agents. To get an understanding of the most important concepts of game theory, a short introduction based on the work of [10] and [11] is given in this section. For a more detailed and complete introduction to game theory, the reader is referred to [10].
Game theory is a mathematical way of modelling the interactions between two or more agents where the outcomes depend on what each agent does and and the study of how these interactions should be structured to lead to good outcomes. Game theory therefore has important applications in many areas such as economics, politics, biology, computer science, philosophy and a variety of other disciplines. It gained recognition during the Second World War, when Oskar Morgenstern and John von

Neumann both published a book on game theory, titled "Theory of Games and Economic Behavior" in 1944 [25]. This book addressed the mathematical analysis of a series of thinking games. A distinction was made between games in which the strategies and the utility factor of the opponent have no effect on finding the best strategies (e.g. chess) and games wherein this factor does have an influence (e.g. poker). John Nash also played a major role in the history of game theory. He was one of the mathematicians who has formalized game theory. The Nash equilibrium, a common solution concept of a non-cooperative game, was named after him.

One of the assumptions underlying game theory is that the players of the game are independent and self-interested. This means they do not actively seek to harm other agents. Instead, each agent has preferences about the state of the world. These preferences are mapped to natural numbers in a function called the utility function. These utility values are a mathematical measure indicating how much an agent likes or dislikes the outcome of the game. Outcomes are the result of a specific combination of a player's strategy. Each combination of a player's strategy is an outcome of the game. A rational, independent and self-interested player prefers outcomes with a higher personal utility value, with no regard for the outcome of other players.

Games can be divided into two types of games: cooperative or non-cooperative. In non-cooperative games, the basic modelling unit is the group of agents. Two or more agents want to maximise their utility and their actions can be affected by other agents' utilities. In the individualistic approach the basic modelling is only one agent. This type of game is referred to as cooperative games. The game modelled in this paper belongs to type of games that are non-cooperative.

Best response and Nash equilibrium

One of the solution concepts in game theory for non-cooperative games that will be used in this paper is a Nash equilibrium. A Nash equilibrium is a subset of outcomes that can be interesting to analyse a game. To define this concept we first introduce the concept of best response. The best response, given an action of the other player, is the action that maximises its pay-off. We define Opt_i as the best response function for player *i*. The best response for player *1* is: $a_1 = Opt_1(a_2)$, given that a_2 is an action of player 2. For a Nash equilibrium each player has a list of actions and each player's action maximises his or her pay-off given the actions of the other players. Nobody has the incentive to independently or unilaterally change his or her action if an equilibrium profile is played. We have a Nash equilibrium for the pair (a_1^*, a_2^*) when $a_1^* = Opt_1(a_2^*)$ and $a_2^* = Opt_2(a_1^*)$.

An example to explain a Nash equilibrium of a two-player game is the prison dilemma. In this game there are two players who are both rational and both of them have committed a crime. 'Rational' means that they want the best for themselves and it is not their purpose to do harm to others. Both are locked in a separate room and they cannot tell in advance to each other what they are going to say. Each of

them can betray the other or they can support each other and remain silent. If a player talks (betray the other prisoner), he will either be imprisoned for 3 years or go free, depending on whether the other talks. If the player is silent, he will either be imprisoned for 5 years or one year, depending on whether the other talks.

Figure 2.1 shows the rewards of the possible actions of the two players. The figure shows that it is advantageous for each player to talk. If prisoner 1 talks and the other prisoner remains silent, prisoner 1 will be free. If prisoner 1 remains silent and the other prisoner talks, prisoner 1 gets five years in prison. If the prisoners cooperate and talk, both of them get three years in prison. If they both remain silent they will both get one year in prison. So this means that talking is the dominant strategy. A dominant strategy is a strategy that is better than all other strategies regardless of what the opponent does. The Nash equilibrium of the game is that they both talk even though it would be better if both prisoners cooperated and choose to stay silent.

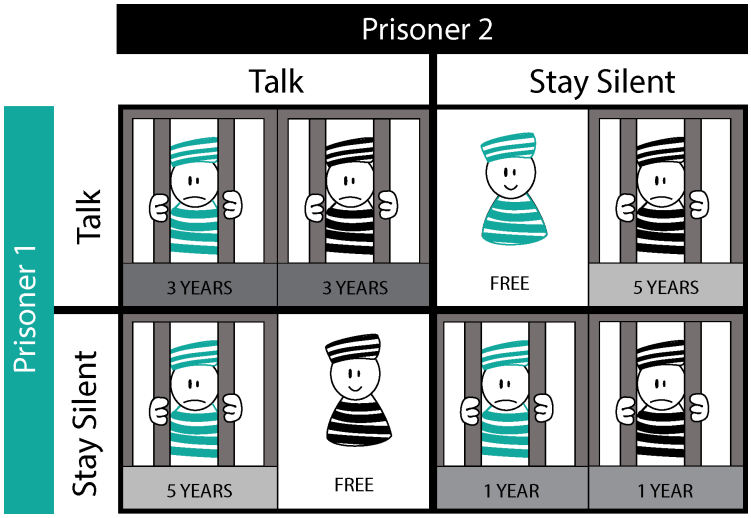


Figure 2.1: Prisoners dilemma: An example from the field of Game Theory. It is a game between two perfectly rational prisoners who do not know what the other one will do. Each prisoner can talk (betray the other prisoner) or remain silent. The resulting sentence is shown below each prisoner.

2.3 The FlipIt Game

FlipIt is a game introduced by van Dijk et al. To understand how to model a FlipIt game with virus propagation it is important to get familiar with the concepts of the normal FlipIt game and its notations. Therefore, we first explain the framework of FlipIt and introduce the most important formulas that will be used throughout the paper.

FlipIt is a two-player game with a shared single resource that the players want to control as much as possible (figure 2.2). The shared resource can be a password, a network or a secret key depending on the setting being modelled. In the remainder of the paper we name the two players the attacker, denoted by the subscript A and the defender, denoted by subscript D.

The game begins at $t = 0$ and continues indefinitely ($t \to \infty$). The time in the game is assumed to be continuous. To get control over the resource, the players i, with $i \in \{A, D\}$, can move (flipping the resource) at any given time. Each move implies a certain cost k_i and can vary for each player. Both players try to minimize their cost. Adding a cost prevents players from moving too frequently.

The unique feature of FlipIt is that every move happens in a stealthy way, meaning that the player does not immediately finds out if the other player (his adversary) has flipped the resource or not. A player only finds out about the state of the game when he moves himself. The goal of the player is to maximise the time that he or she has control over the resource while minimizing the total cost of the moves. A move can also result in a "wasted move", called a flop. It may happen that the resource was already under control of the player. If the player moves when he or she has already control over the resource, he or she would have wasted a move since this does not result in a change of ownership, so the cost is wasted.

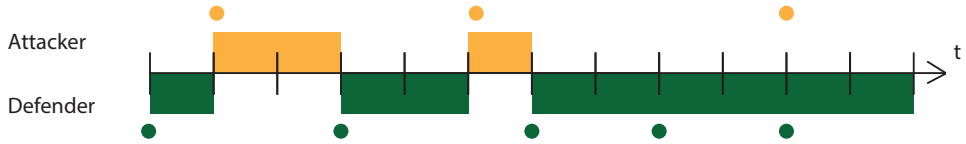


Figure 2.2: A representation of a FlipIt game where both players are playing periodically. Every move or flip is indicated by a green (dark grey) or orange (light grey) circle. The defender is represented in green and the attacker is represented in orange. The green and orange rectangles represent which player is in control of the resource.

We denote the state of the resource as a time-dependent variable $C = C_i(t)$. $C_D(t)$ is 1 if the game is under control by the defender and 0 if the game is under control by the attacker. Reversely, $C_A(t)$ is 1 if the game is under control by the attacker and 0 if under control by the defender. So, $C_A(t) = 1 - C_D(t)$. The game starts with the defender being in control: $C_D(0) = 1$.

The total gain for player i is equal to the total amount of time that player i has owned the resource from the beginning of the game up to time t:

$$G_i(t) = \int_0^t C_i(x)dx. \tag{2.1}$$

The gain of the attacker and the defender always sums up to t:

$$G_D(t) + G_A(t) = t \tag{2.2}$$

The average gain rate of player i is defined as:

$$\gamma_i(t) = G_i(t)/t. \tag{2.3}$$

And thus for all $t > 0$:

$$\gamma_D(t) + \gamma_A(t) = 1 \tag{2.4}$$

The players receive a benefit equal to the time units they were in possession of the resource minus the cost of making their moves.

$$\beta_i(t) = \gamma_i(t) - k_i\alpha_i. \tag{2.5}$$

where $\beta_i(t)$ denote player's i average benefit rate, k_i denotes the cost for player i and α_i defines the average move rate by player i up to time t with $n_i(t)$ the amount of moves made by player i up to time t:

$$\alpha_i(t) = \frac{n_i(t)}{t} \tag{2.6}$$

In a given game, the asymptotic benefit rate (or simply benefit) will be defined as the *lim inf* of the average benefit because time t will increase to infinity and the average benefit may not have limiting values.

$$\beta_i(t) = \lim_{t\to\infty} inf\beta_i(t) \tag{2.7}$$

Strategies

Because the players move in a stealthy way, there are different types of feedback a player can get by flipping the resource. These types of feedback can be divided into two groups of strategies. The non-adaptive strategies and the adaptive strategies. These are described in table 2.1.

If there is no feedback for either player, the player will play in the same manner against every opponent. The strategy is then called *non-adaptive* because the playing strategy is not dependent on the opponent's movements. An interesting subclass of the non-adaptive strategies are the Renewal strategies where the time intervals between two consecutive moves are generated by a renewal process. The length between two consecutive moves are independent and identically distributed random variables chosen from a density formula *f*. This renewal process means that the strategies are renewed after each move: the interval until the next move only depends on the current move time and not on previous history. An example of such renewal

Categories	**Classes of Strategies**
Non-adaptive (NA)	Renewal
	- Periodic
	- Exponential
	General non-adaptive
Adaptive (AD)	Last move (LM)
	Full History (FH)

Table 2.1: Hierarchy of Classes of strategies in FlipIt

strategy is the periodic strategy where the time between two consecutive moves of the players is a fixed interval. An exponential strategy is a renewal strategy in which the interval between two consecutive moves is exponentially distributed.

If the players receive feedback, a player can adapt his strategy to the information received about the opponent's moves. This strategy is called an *adaptive* strategy. Depending on the amount of information received, two subclasses of adaptive strategies can be identified. The Last Move (LM) strategies represent the class where, whenever a player flips, he will find out the exact moment that the opponent moved the last time. In the second class, called Full History (FH), whenever a player flips he will find out the whole history of the opponent's moves.

Results of the FlipIt game

The study of the different strategies by means of the FlipIt framework allows to derive a number of interesting results [24]:

- a periodic strategy strongly dominates the other renewal strategies if the opponent has a periodic or non-arithmetic renewal strategy. This means that it is a good choice for the opponent to play periodically against a player with a non-adaptive strategy;

- periodic games are disadvantageous against players following a LM adaptive strategy. The opponent can observe the exact time of the player's next move and play immediately afterwards. If the costs are from the same magnitude, the opponent can keep the control over the resource with little interrupts from the other player;

- a player facing an LM opponent and with a cost much lower than the opponent has two options. The first option is to move with a periodic rate that is fast enough he'll force the opponent to drop out. The other option is to play with a randomized strategy, such that the opponent cannot learn any information regarding the next move of the player;

- the best defence strategy is to play fast, to make the opponent drop out of the game. To be able to move fast, the player has to make sure that the cost of moving is much less than the opponents moves.

2.4 Related Work on Extensions to FlipIt

Various possible ways to extend FlipIt have already been proposed. Laszka et al. made a lot of additions and extensions to the original game of FlipIt. For instance Laszka et al. extended the basic FlipIt game to multiple resources. The rationale is that for compromising a system in real life, more than just one resource needs to be taken over. For example, gaining access to deeper layers of a system may require breaking several passwords. This model is called FlipThem [7]. Laszka et al. uses two ways to flip the multiple resources: the AND and the OR control model. In the AND model the attacker only controls the system if he controls all the resources of the system, whereas in the OR model the attacker only needs to compromise one resource to be in control of the entire system.

Another addition of Laszka et al. [8] to the game of FlipIt is extending the game to also consider non-targeted attacks by non-strategic players. In this game the defender tries to maintain control over the resource that is subjected to both targeted and non-targeted attacks. Non-targeted attacks can include phishing, while targeted attacks may include threats delivered through zero day attack vulnerabilities.
One of the last important additions from Laszka et al. [9] is to consider a game with targeted and non-targeted attacks where the moves made by the attacker do not succeed immediately. This approach is similar to what will be done in this paper, but nevertheless has some major differences. Firstly, in Laszka's paper, the moves by the attacker are still covert but the moves made by the defender are known to the attacker. This means that the attacker knows when the defender plays and can change its strategy depending on the moves of the defender. Our motivation for a defender with stealthy moves is that it can not be assumed that every attacker can receive feedback from an APT. Some ATPs are designed only to cause harm on the systems that are infected e.g. the *Stuxnetworm* that was developed to destroy nuclear reactors. The *Stuxnetworm* was resident on a isolated network, meaning that there was no way to connect to the internet. Another example is the use of honeypots. Honeypots emulate services or create multiple instances of real operating systems and can pretend to have sensitive information. Honeypots do not detect all malicious attacks so the attacker can stay unnoticed and can still provide information as feedback. By providing false information through honeypots, the defender can also remain stealthy. The second difference is that even though both the targeted and non-targeted attacks do not succeed immediately, the delay is determined differently. For the targeted attack the time till it succeeds is given by an exponential distributed random variable with a known rate. The non-targeted attacks are modelled as a single attacker and the time until it succeeds is given by a Poisson process. In this paper the delay is given by one parameter, which can be the result of any virus

propagation model. The third and last difference is that the paper of Laszka has multiple attackers who try to find the best strategy of the defender against both targeted and non-targeted attacks. The conclusion of Laszka's paper is that the optimal strategy for the defender is moving periodically.

FlipIt also has been applied to several cases in computer security. Researchers explored different applications of FlipIt for real-world problems, like password reset policies, VM refresh, cloud auditing and key rotation [1].
Other authors used the FlipIt game to apply it to a specific scenario. To be able to use the FlipIt game, modifications were required for the FlipIt model. One of the scenarios by Pham [15] was to find out whether a resource was compromised or not by the attacker. This could be verified by the defender, who has an extra move "test" beside the flip move. The basic idea is to test with an extra action if the resource has been compromised or not. This move also involves an extra cost.
A three-player game has also been investigated where the FlipIt framework of two players is extended by another player. This player represents an insider that trades value information with the attacker [3].

Finally, researchers have also investigated the behaviour of humans playing FlipIt. A. Nochenson and Grossklags [14] investigated how people really act when given temporal decisions. They found out that the results improve over time but that they are dependent on gender, age, and other individual difference variables. The result also shows that the participants perform generally better when they have more information about the strategy of the opponent, which is a computerized player. Reitter et al. [17] extended the work of A. Nochenson and Grossklags to include various visual presentation modalities for the available feedback during the investigation.

Chapter 3

FlipIt with Propagation Delay

The FlipIt game with propagation delay considers a game where the moves of the attacker are not instantaneous. This corresponds to APTs which use malware - viruses, worms or trojan horses - to perform attacks, as the malware needs time to propagate. A virus, for example, can be dropped on a network but it only compromises the whole network if every node in the network is infected. So there is a certain delay between the moment of attack and the moment the attacker has control over the resource, called the propagation delay. The basic FlipIt game does not take this propagation delay into account. This chapter explains how the FlipIt game with propagation delay can be modelled. Section 3.1 explains the difference between a basic FlipIt game and a FlipIt game with propagation delay. The last section 3.2 derives a formula to calculate the benefit for a FlipIt game with propagation delay. In the next Chapter, this benefit formula will be used to determine what the best defence and attack strategies are.

3.1 Difference between FlipIt with and without Propagation Delay

The following paragraphs will list the required adaptations to model the basic FlipIt game with propagation delay.

3.1.1 Single resource

The basic FlipIt game consists of a single resource. To represent the security problem of the propagation of an APT in a network, the adapted game defines its single resource as a computer network with multiple nodes. One of the players, the defender, will try to defend his network. The defender will do this by flipping the nodes of the network. The attacker on the other hand will try to infect all the nodes in the network. The attacker will do this by dropping a virus on a node on the network. The virus will then spread itself and infect other nodes in the network.
By defining the resource as a network with multiple nodes it is possible to increase

the number of possible actions by the defender or attacker. These actions will be explained in the following subsection.

3.1.2 Actions of the players

The network is composed of multiple nodes. The defender can choose to flip one node, a subset of nodes or all nodes of the network. The analyses in this paper is based on the assumption that the defender flips all the nodes in the network every time he plays. This action can be extended to flipping only a subset of the nodes or a single node of the network. This extension has already been investigated in the context of installing anti-virus systems on specific nodes in a graph and investigating how this affects the spreading of malware. In chapter 5, when discussing future work, the use of a matrix model is discussed to determine the probability that a node is infected by malware. A practical example of a flip of the defender may consist of patching, formatting, disk-wiping or unplugging a computer on the network.
The attacker only has one action: sending different kinds of malware to a computer network. Every time the attacker flips the resource, he does this by sending a new kind of malware e.g virus to the system. The attacker does not send the same malware again, because if the action of the defender, e.g. consists of patching, the malware will be unable to penetrate the system. The node can be a targeted node or a random node. If the attacker has knowledge about the topology of the network, the attacker can choose to target a specific node. Most likely, this will be the node that can infect all other nodes in a minimum timespan.

3.1.3 Immediate effect of the move

The time that it takes for a piece of malware to infect every node (or a sufficient number of nodes) will be denoted as a propagation delay variable d (called 'delay' for short in the remainder of this paper). If we want to measure how long it takes for the malware to infect all the nodes in the network, we have to calculate the shortest path from the first infected node to the farthest node. Rather than denoting the time needed for infecting *all* the nodes, the variable d can also be used to denote the time needed to infect *a sufficient number* of nodes.
The moves of the defender are immediate. It is assumed that if a defender tries to clean the network that this will happen without a delay. The defender can de-plug the computer from the network, push a security update to all pc's, or even format a pc.

3.1.4 Stealth character of the move

Moves in the basic FlipIt game are stealthy or covert and not immediately detected by the other player. The attacker's moves are stealthy because we want to model a scenario where a computer network is attacked by an APT. The main characteristic of an APT is that the attack is stealthy. Depending on how the attacker has set up his attack, and depending on whether or not the network is connected with the

Internet, the moves of the defender are stealthy or not stealthy. For example, if the attacker launches an APT only to harm the system and not to receive feedback, the moves of the defender are stealthy. If the attacker launches an APT to steal sensitive information, the moves of the defender are non stealthy because if the defender takes actions the attacker can see that he does not receive information any more.
In this paper we assume that both moves of the attacker and the defender are stealthy. Our motivation for a defender with stealthy moves is that it can not be assumed that every attacker can receive feedback from an APT. Some APTs can be resident on a isolated network (e.g. the *Stuxnetworm*), meaning that there is no way to connect to the Internet.
Even when the APT is launched to steal sensitive information the defender can set-up a honey pot, making the attacker believe that he is stealing sensitive information, but the information in the honeypot is all fake. Honeypots emulate services or creates multiple instances of real operating systems and can pretend to have sensitive information. While the analysis in this paper is limited to immediate effect of the defender's moves, the case where the moves of the defender are non stealthy has already been investigated by Laszka [9].

3.1.5 Cost associated with the move

The defender will defend every computer-controlled device in a network. In order to clean the network, the defender can patch the systems with the latest security bulletins, reinstall the software or even format the computer. All these different actions can imply other costs. The cost of patching a system, for example, will most likely be smaller than replacing a computer.
The cost of the attacker depends on the complexity of the vulnerabilities exploited and how difficult it was to program the malware.

3.1.6 Strategies

The formal definition of the adapted FlipIt game starts from the model of the non-adaptive continuous basic FlipIt game where players use a periodic strategy with a random phase.

No feedback (non-adaptive)
Non-adaptive strategies are chosen because of the assumption that both players will receive no feedback during the game. This is motivated by the fact that the identity of the attackers and their attack strategy is rarely known to the defenders. In the case of the attacker, if an attacker wants to have feedback, receiving feedback may compromise the desire to stay undetected. To be able to get feedback, the malware needs to make a connection with the attacker to provide this feedback. This may for example cause extra network traffic, which can be detected by the defender.

Periodic strategy
The choice of periodic strategy is motivated by the assumption that in most organi-

sations, the defence strategy is to periodically defend the network. This is true for the example of patching. Companies like Microsoft and Google release their security patches at fixed intervals, so companies will apply these patches as they come out. Microsoft security bulletins are released, for example, on the Second Tuesday of each month. [12]

In the basic FlipIt game [24] the authors van Dijk et al. concluded that a periodic strategy is the dominant strategy against all other renewal strategies if the opponent uses a periodic or non-arithmetic renewal strategy. So regardless of whether the defender chooses to play periodically or not, it's a good choice for the attacker to play periodically as well.

The paper from Laszka [9], which is similar to this one, also concluded that a periodic strategy is the dominant strategy against all other non-adaptive strategies. Further research can investigate the effect of relaxing this assumption.

3.2 Formalization of the Periodic Game with Propagation Delay

A Periodic strategy is a non-adaptive renewal strategy where the time intervals between consecutive moves are a fixed period, denoted by δ. It has a random phase, that is chosen uniformly and randomly in the interval $[0, \delta]$ for the first move. The average rate of play of a player is denoted by $\alpha_i = \frac{1}{\delta_i}$. Given below is a list of symbols that will be used throughout the paper. Figure 3.1 clarifies the main symbols.

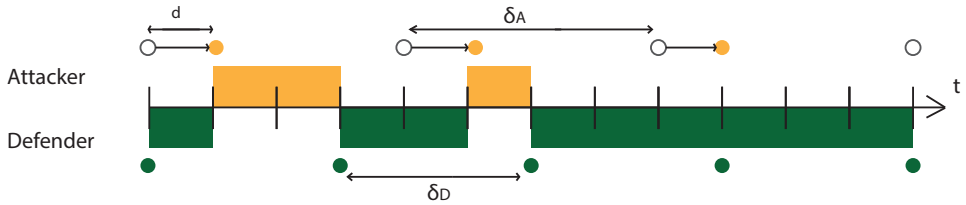


Figure 3.1: Formalization of a FlipIt game with propagation delay: A representation of a FlipIt game where both players are playing periodically. Every move or flip is indicated by a green or orange circle, respectively dark gray and light grey. The defender is represented in green and plays with a period of δ_D. The flip of the attacker is represented by a white circle, but because there is a delay d, the attacker only controls the resource after time d represented by an orange circle. The attacker plays with a period of δ_A. The green and orange rectangles represent the amount of time the respective player is in control of the resource.

i: Denotes the player. D denotes the defender, and A denotes the attacker which

differs from the notation in [24], where the defender is denoted by the subscript *0* and the attacker by the subscript *1*.

δ_i: The length of the interval between two consecutive moves of player i.

α_i: The average flip rate of player i, given by $\alpha_i = 1/\delta_i$.

k_i: The cost of player i's moves.

d: The delay caused by the virus propagation.

$G_i(t)$: The total gain of player i, which is the amount of time player i is in control over the resource up to time t.

γ_i: The average gain rate of player i, defined as $G_i(t)/t$

β_i: The average benefit rate up to time t, defined as $\beta_i = \gamma_i - k_i\alpha_i$.

opt_i: The optimum function for player i.

The adaptation of the FlipIt model starts from the assumption that, when an attacker attacks at time t, he doesn't get immediate control over the resource, but he only gains control at time $t + d$, with d denoting the time needed to infect a sufficient number of (or all) nodes. If the defender flips the network before the period d has elapsed (so, somewhere between t and $t + d$), then the attacker will never gain full control over the resource. (See figure 3.2 at points 3 and 4). This implies that the mathematical formulas for gain and benefit need to be adapted to the fact that the attacker loses part of his benefit because of this delay. In the remainder of this paper, we will adapt the formalization of the FlipIt game using the delay variable d.

We apply the same definition for players that play at the same time as in the original FlipIt game. As the time in the game is continuous, the probability of this happening is close to zero, but if it happens the moves cancel each other out and no change of control happens (see figure 3.2 point 3).

The formalization is split three cases. Similarly as in the original FlipIt game, there is the case where the defender plays at least as fast as the attacker, and the case where the attacker plays at least as fast as the defender. For each of these cases, the benefit formula of the basic case without delay is presented first, and subsequently the delay is introduced. The third case is a special case where the period of the defender is smaller than the delay.

In the original model of FlipIt, the authors express the formulas in terms of α_D and α_A. However, when introducing the delay, some formulas become much simpler when expressed in terms of δ_D and δ_A. Therefore in the remainder of this paper, we will formulate the model using both α_i and δ_i, depending on which variable gives the simplest representation of the formulas.

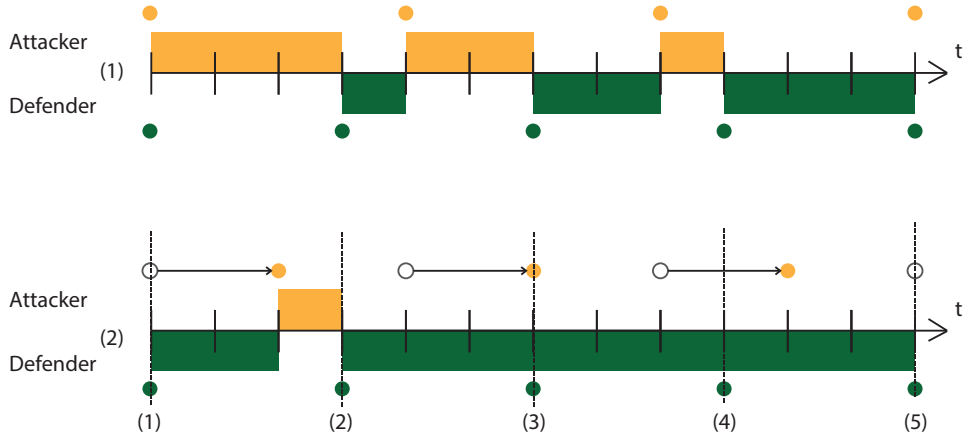


Figure 3.2: The first game is the basic FlipIt game. The second is the FlipIt game with a delay. During the first flip of the attacker, the defender moves after the delay (point 2), resulting in the attacker gaining control over the resource, until the defender flips. During the second flip of the attacker, the defender flips at time t+d, causing the defender to retain control over the resource before the attacker can gain control. During the third and final flip of the attacker, the defender flips during time t+d, causing the attacker to never gain control over the resource.

Case 0: $\delta_D \leq d$ (The defender plays with a period smaller than the delay)

When the delay d is bigger than δ_A, the attacker would play again before the delay has finished. This would seemingly result in a gain for the defender that is always 1, but this is not always true. Assume for example that an attacker plays with an interval of 3 time units, that the delay is equal to 4 time units and that the defender only plays every 8 time units. This situation is represented in figure 3.3. Since the delay is shorter than the period of the defender, the attacker takes control of the resource once the delay has elapsed, until the defender plays.

However, if the delay is larger than or equal to the period of the defender, the attacker will never gain control over the resource. The defender will always flip before the delay is over. This situation is represented in figure 3.4, where the attacker plays with an interval of 5 time units, the delay is equal to 6 time units and the defender plays with an interval of 4 time units.
When $\delta_D \leq d$, it does not matter for the attacker whether he plays faster or slower than the defender. For this case the benefit of the defender and the attacker are as follows:

$$\beta_D(\delta_D, \delta_A) = 1 - \frac{k_D}{\delta_D} \tag{3.1}$$

$$\beta_A(\delta_D, \delta_A) = -\frac{k_A}{\delta_A} \tag{3.2}$$

For the remainder of the cases we assume that $\delta_D \geq d$.

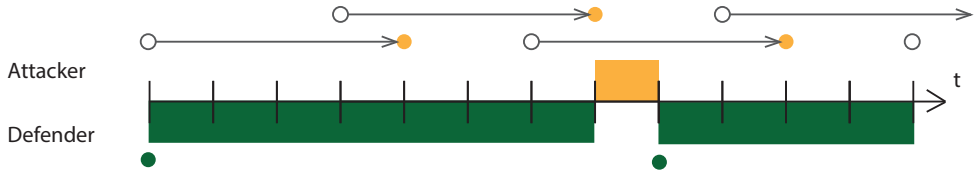


Figure 3.3: FlipIt with delay propagation where $\delta_D \geq d \geq \delta_A$.

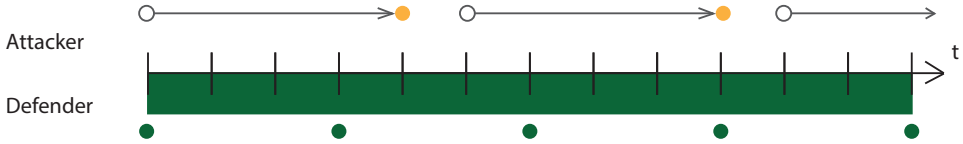


Figure 3.4: FlipIt with delay propagation where $d \geq \delta_D$

Case 1: $\delta_D \leq \delta_A$ (The defender plays at least as fast as the attacker.)

Let $r = \frac{\delta_D}{\delta_A}$. The intervals between two consecutive defender's moves have length δ_D. Consider a given defender move interval. The probability over the attacker's phase selection that the attacker moves in this interval is r. Given that the attacker moves within the interval, he moves exactly once within the interval (since $\delta_D \leq \delta_A$) and his move is distributed uniformly at random within this interval.

The expected period of attacker control within the interval as the moment on which the attacker gains control is uniformly distributed over the defender's interval. On average the attacker will gain control at time r/2. So the expected gain is equal to the remainder of the defender interval, i.e. r/2, without considering the delay by a virus. Therefore the benefit for the attacker, without considering the delay, can be expressed as follows:

$$\beta_A(\delta_D, \delta_A) = \frac{r}{2} - k_A\alpha_A = \frac{\delta_D}{2\delta_A} - k_A\alpha_A$$

Correspondingly, the benefit for the defender can be expressed as:

$$\beta_D(\delta_D, \delta_A) = 1 - \frac{r}{2} - k_D\alpha_D = 1 - \frac{\delta_D}{2\delta_A} - k_D\alpha_D$$

However, because of the delay required for virus propagation, the maximal time of control is reduced to $\delta_D - d$, see figure 3.5. While the probability that the attacker will move in the interval of the defender is still r, the gain will not be half of the interval. Indeed, if the attacker plays after $\delta_D - d$, given the delay d, he will never gain control in that interval (see figure 3.6). The probability that the attacker plays early enough is $\frac{\delta_D - d}{\delta_D}$, giving the attacker an average gain of $\frac{\delta_D - d}{2}$ (the average remainder of the defender interval after the attacker flipped). If the attacker moves

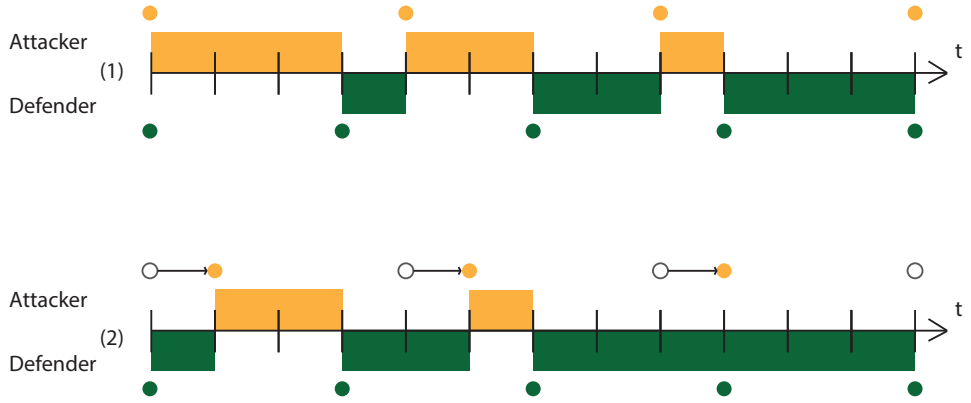


Figure 3.5: Case 1: Difference between a basic FlipIt game and a FlipIt game with a delay. Case (1) is the FlipIt game without a propagation delay and case (2) is with a propagation delay. The delay is denoted with an arrow. The attacker is only in control when the circle becomes orange (light grey).

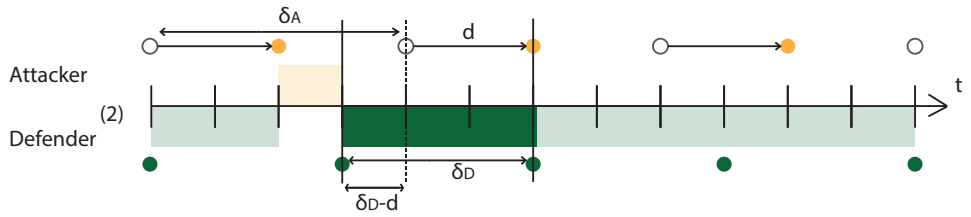


Figure 3.6: Attacker playing to late. If the attacker enters the defender's interval after $\delta_D - d$, he can not get in control in that interval.

after the period of $\delta_D - d$, the gain of the attacker will be zero. The probability that this happens is $\frac{d}{\delta_D}$. Looking at one interval of the defender, the average gain rate of the attacker can thus be expressed as follows:

$$\gamma_A(\delta_D, \delta_A) = \frac{1}{\delta_D}[\frac{\delta_D}{\delta_A} \cdot [\frac{\delta_D - d}{\delta_D} \cdot \frac{\delta_D - d}{2} + \frac{d}{\delta_D} \cdot 0]]$$

As the formula above is valid for each defender interval, the average gain rate over the entire game is:

$$\gamma_A(\delta_D, \delta_A) = \frac{(\delta_D - d)^2}{2\delta_D\delta_A}$$

To find the benefit, the cost of moving is subtracted from the average gain.

$$\beta_A(\delta_D, \delta_A) = \frac{(\delta_D - d)^2}{2\delta_D\delta_A} - \frac{k_A}{\delta_A} \tag{3.3}$$

The benefit of the defender is then as follows:

$$\beta_D(\delta_D, \delta_A) = 1 - \frac{(\delta_D - d)^2}{2 \cdot \delta_D\delta_A} - \frac{k_D}{\delta_D} \tag{3.4}$$

For this case where d=0, this equals the formula of the original FlipIt game [24] [p675].
For the border case, when $d = \delta_D$, the benefit of the defender and the attacker result in the same benefits as in case 0.

Case 2: $\delta_A \leq \delta_D$ (The attacker plays at least as fast as the defender.)

First let $r = \frac{\delta_D}{\delta_A}$. The intervals between two consecutive attacker's moves have length δ_A. Consider a given attackers move interval. The probability over the attacker's phase selection that the defender moves in this interval is $\frac{\delta_A}{\delta_D} = (1/r)$. Given that the defender moves within the interval of the attacker, he moves exactly once within this interval (since $\delta_A \leq \delta_D$) and his move is distributed uniformly at random.

A similar analysis as in case 1 for a FlipIt game without a propagation delay yields the following benefits:

$$\beta_D(\delta_D, \delta_A) = \frac{1}{2r} - k_D\alpha_D = \frac{\delta_A}{2\delta_D} - \frac{k_D}{\delta_D}$$

$$\beta_A(\delta_D, \delta_A) = 1 - \frac{1}{2r} - k_A\alpha_A = 1 - \frac{\delta_A}{2\delta_D} - \frac{k_A}{\delta_A}$$

An intuitive solution for the case with a virus would be to subtract the benefit of the attacker received in each interval with the delay similarly as in case 1. This would yield the following formula:

$$\beta_A(\delta_D, \delta_A) = \frac{(\delta_A - d)^2}{2\delta_A\delta_D} - \frac{k_D}{\delta_A}$$

This however results in an overestimation. The reason this formula overestimates the benefit of the attacker is that it assumes that the defender is always in control during the delay. However, if the attacker was in control in the previous interval, then he continues to be in control during the period of the delay, see figure 3.7 situations 5 to 7. This means that the average benefit formulas for this case cannot be derived from one interval only; what happens in the previous interval must be taken into account.

We know that the defender's moves are instantaneous. Therefore, it is easier to calculate the benefit of the defender. Because the defender moves more slowly than the attacker we know that if the defender moves during the interval of the attacker, he only moves once within this interval.

The defender will move during the interval of the attacker with a probability of $\frac{\delta_A}{\delta_D}$. If this happens, the defender will be in control for the remainder of this interval.

In the next interval the attacker will have to regain control, meaning that during the delay, the defender stays in control, see figure 3.7 cases (1) to (4). The defender will keep the control over the resource in the next interval over a period of the delay, namely d.

Consider a timespan $\delta_A + d$, representing the attacker's interval followed by the delay period in his next interval. If we assume that $\delta_A + d < \delta_D$, we can infer that the defender will never move twice during this timespan. Because $d + \delta_A \leq \delta_D$ the next move of the defender in this second interval will never occur during the delay, meaning that the entire delay can be considered as an extra benefit resulting of a play in the previous interval. So, every time the defender plays, he will get an average gain of $\frac{\delta_A}{2}$ in the interval where he plays and in the next interval will always receive an extra gain of d, yielding a total average gain per interval of $\frac{(d + \frac{\delta_A}{2})}{\delta_A}$

For the remainder of the analyses we consider two cases, Case 2.a and Case 2.b, depending on whether the delay is shorter or longer than the difference between the attacker's and the defender's period.

Case 2.a: $\delta_D \geq d + \delta_A \geq \delta_A$

In this case the delay will never be counted twice in the defender's benefit formula. To determine the total gain rate of the defender we need to know the probability that the defender will move during an interval and what the average time is that the defender controls the resource. Given that if the defender moves he will always benefit entirely from a period of delay in the next interval of the attacker, his total gain is $\frac{(d + \frac{\delta_A}{2})}{\delta_A}$ in one interval (as previously calculated). The total gain rate of the defender is then the probability that the defender will move during an interval of the attacker multiplied by the total average gain per interval:

$$\gamma_D(\delta_D, \delta_A) = \frac{\delta_A}{\delta_D} \cdot \frac{(d + \frac{\delta_A}{2})}{\delta_A}$$

$$\gamma_D(\delta_D, \delta_A) = \frac{\delta_A}{2\delta_D} + \frac{d}{\delta_D}$$

This yields the following benefit formula:

$$\beta_D(\delta_D, \delta_A) = \frac{\delta_A}{2\delta_D} + \frac{d}{\delta_D} - \frac{k_D}{\delta_D} \tag{3.5}$$

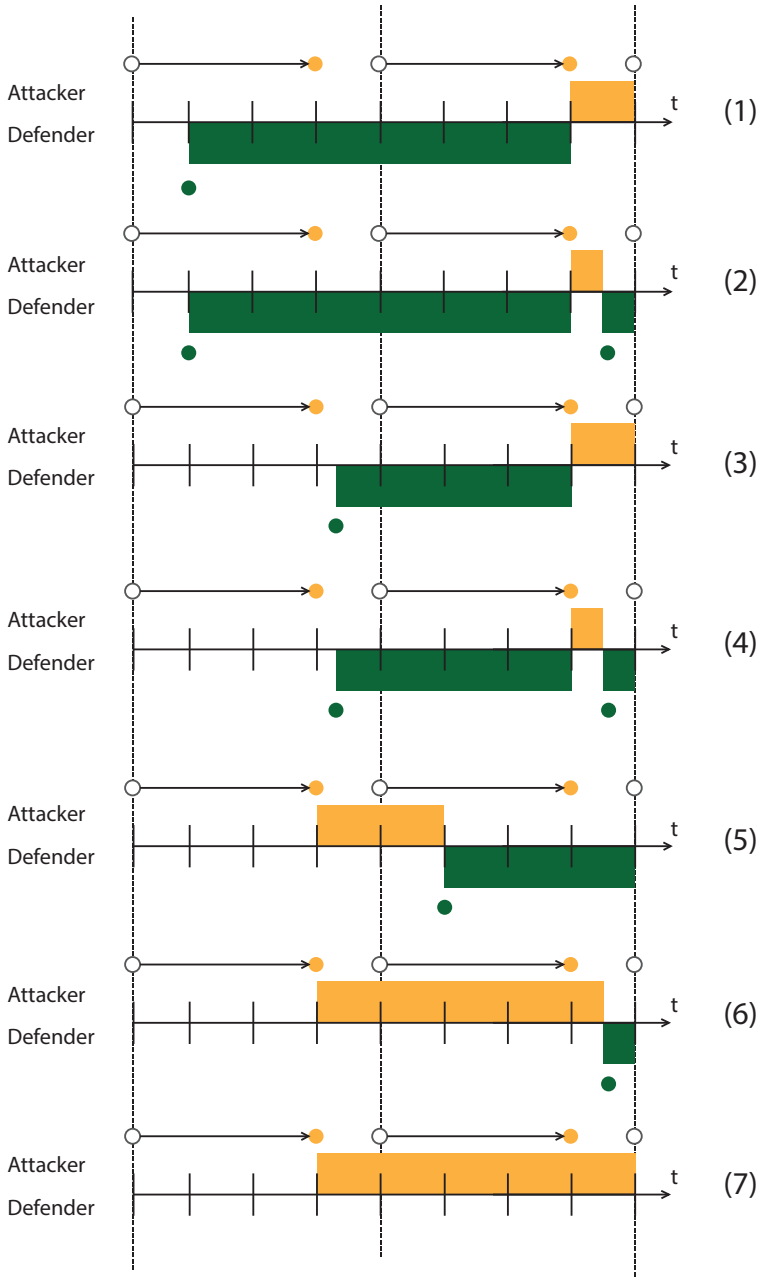


Figure 3.7: All possible cases for the attacker and the defender in Case 2.A where $d+\delta_A \leq \delta_D$. As can be seen in cases (1) to (4), the defender will have control during a period of d over the resource in the next interval when the defender has flipped in the previous interval.

The benefit for the attacker will be as follows:

$$\beta_A(\delta_D, \delta_A) = 1 - \frac{\delta_A}{2\delta_D} - \frac{d}{\delta_D} - \frac{k_A}{\delta_A} \tag{3.6}$$

It is crucial that δ_D is at least as large as $d + \delta_A$. If not, the defender can move during the delay in the interval following the interval in which the defender already moved. This would result in an overlap between the average gain of $\frac{\delta_A}{2} + d$ and the delay. The above benefit formula would then include too much gain for the defender: the potential overlap during the delay would be counted twice. See figure 3.8.

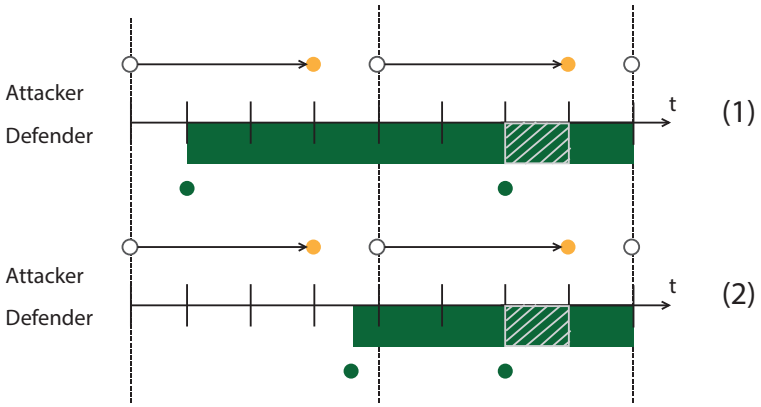


Figure 3.8: Cases where the delay would be counted twice.

Case 2.b: $d + \delta_A \geq \delta_D \geq \delta_A$

To obtain the formula in case of a too long delay, we therefore need to subtract this overlapping gain from the above formula. Since $\delta_D \geq \delta_A$, if the defender enters the interval immediately after the attacker has played, the defender cannot have played in the previous interval. In that case, there is no overlap. So the problem of the overlap only appears if the defenders enters late enough and thus only the last part of the delay is subject to overlap. The larger the difference between the interval of the defender and the attacker, the smaller the risk of overlap. Concretely, only the last part of length $d-(\delta_D-\delta_A)$ is subject to overlap. Hence, the probability of overlap is $\frac{d-(\delta_D-\delta_A)}{\delta_D}$ and the average gain will be half of this interval: $\frac{d-(\delta_D-\delta_A)}{2}$. The gain rate to be subtracted is therefore:

$$\frac{1}{\delta_A} \cdot \frac{d - (\delta_D - \delta_A)}{\delta_D} \cdot \frac{d - (\delta_D - \delta_A)}{2}$$

The total gain rate for the defender is obtained by subtracting this term from the gain rate of case a:

$$\gamma_D(\delta_D, \delta_A) = \frac{\delta_A}{\delta_D} \cdot \frac{(d + \frac{\delta_A}{2})}{\delta_A} - \frac{(d - (\delta_D - \delta_A))^2}{2\delta_D\delta_A}$$

$$\gamma_D(\delta_D, \delta_A) = \frac{\delta_A}{2\delta_D} + \frac{d}{\delta_D} - \frac{(d - (\delta_D - \delta_A))^2}{2\delta_D\delta_A}$$

This yields in the following benefit formula:

$$\beta_D(\delta_D, \delta_A) = \frac{\delta_A}{2\delta_D} + \frac{d}{\delta_D} - \frac{k_D}{\delta_D} - \frac{(d - (\delta_D - \delta_A))^2}{2\delta_D\delta_A} \tag{3.7}$$

Consequently, the benefit for the attacker will be:

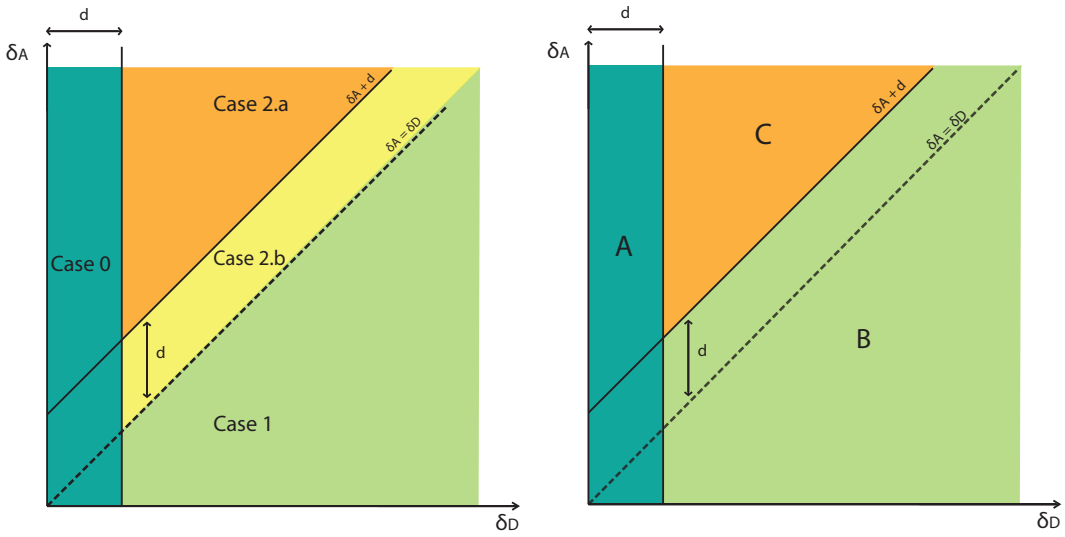
$$\beta_A(\delta_D, \delta_A) = 1 - \frac{\delta_A}{2\delta_D} - \frac{d}{\delta_D} - \frac{k_A}{\delta_A} + \frac{(d - (\delta_D - \delta_A))^2}{2\delta_D\delta_A} \tag{3.8}$$

3.3 Summary

We calculated the benefit for three cases: when δ_D is smaller or equal to d, when δ_D is smaller or equal to δ_A and when δ_D is larger or equal to δ_A. The benefit formulas of case 1 and case 2.b result in the same formula. Consequently, these two formulas can be brought together by adapting the border conditions. The first figure in figure 3.9 shows all the cases and it can be seen that the domain of case 1 and case 2.b can be brought together. This can be seen as the second figure in 3.9.

The benefit functions can be represented as piecewise functions for each case, with case 1 and case 2.b together:
The benefit formula for the defender is as follows:

$$\beta_D(\delta_D, \delta_A) = \begin{cases} 1 - \frac{k_D}{\delta_D}, & \delta_D \leq d \\ 1 - \frac{(\delta_D - d)^2}{2\delta_D\delta_A} - \frac{k_D}{\delta_D}, & d \leq \delta_D \leq d + \delta_A \\ \frac{\delta_A}{2\delta_D} + \frac{d}{\delta_D} - \frac{k_D}{\delta_D} & \delta_D \geq d + \delta_A \end{cases}$$



Case 0 : $\delta_D \leq d$
Case 1 : $\delta_D \leq \delta_A$ & $d \leq \delta_D$
Case 2.a : $\delta_A \leq \delta_A + d \leq \delta_D$ & $d \leq \delta_D$
Case 2.b : $\delta_A \leq \delta_D \leq \delta_A + d$ & $d \leq \delta_D$

A : $\delta_D \leq d$
B : $d \leq \delta_D \leq \delta_A + d$
C : $\delta_A + d \leq \delta_D$

Figure 3.9: The first figure is the representation of the cases where the benefit functions are calculated. The second figure is the representation of the new domain of piece B which is a merge of case 2.b and case 1.

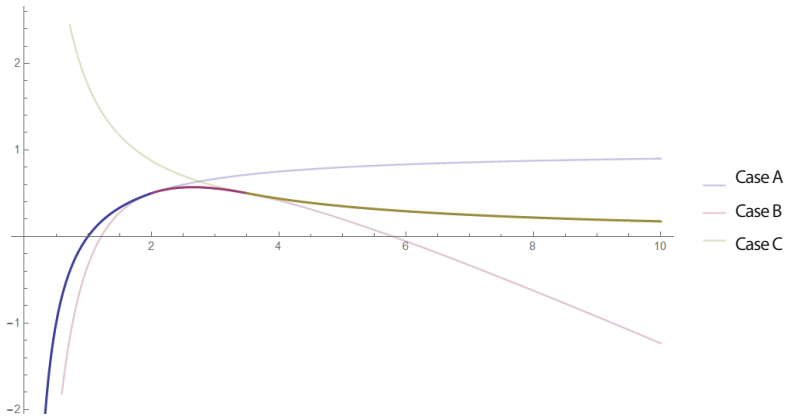


Figure 3.10: The benefit function of the defender for a cost $k_D = 1$, $d = 2$ and $\delta_A = 1.5$. The three pieces match with the three colours.

The benefit formula for the attacker is as follows:

$$\beta_A(\delta_D, \delta_A) = \begin{cases} -\dfrac{k_A}{\delta_A}, & \delta_D \leq d \\ \dfrac{(\delta_D - d)^2}{2\delta_D\delta_A} - \dfrac{k_A}{\delta_A}, & d \leq \delta_D \leq d + \delta_A \\ 1 - \dfrac{\delta_A}{2\delta_D} - \dfrac{d}{\delta_D} - \dfrac{k_A}{\delta_A} & \delta_D \geq d + \delta_A \end{cases}$$

To be able to show the continuity of the benefit function, piece 2 and piece 3 are switched and the boder conditions are reformulated.

$$\beta_D(\delta_D, \delta_A) = \begin{cases} 1 - \dfrac{k_D}{\delta_D}, & \delta_D \leq d \\ \dfrac{\delta_A}{2\delta_D} + \dfrac{d}{\delta_D} - \dfrac{k_D}{\delta_D} & \delta_A \leq \delta_D - d \\ 1 - \dfrac{(\delta_D - d)^2}{2\delta_D\delta_A} - \dfrac{k_D}{\delta_D}, & \delta_D - d \leq \delta_A \end{cases}$$

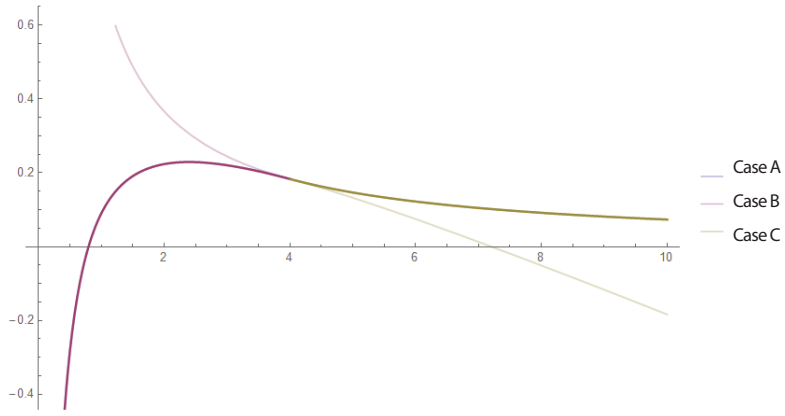


Figure 3.11: The benefit function of the attacker for a cost $k_A = 0.5$, $d = 3$ and $\delta_D = 7$.

The piecewise functions are also continuous functions. Filling in the border cases results in the same formula. This can be seen on both figures 3.10 and 3.11 that give respectively a visual representation of the benefit function of the defender and the benefit function of the attacker. The thick pieces indicate the cases for which each

piece of the benefit function holds. For the benefit of the attacker case B and case C correspond respectively with piece 2 and piece 3 of the reformulated benefit function of the attacker.

Chapter 4

Optimal Defence and Attack Strategies

In this chapter we are interested in finding the optimal defence and attack strategies. Ultimately, the optimal strategies can allow the determination of the Nash equilibria of the game.

Nash equilibria are points with the property that neither player benefits by deviating in isolation from the equilibrium. We can compute Nash equilibria for the periodic game as an intersection point of curves opt_D and opt_A.
More formally, a Nash equilibrium for the periodic game is a point (δ_D^*, δ_A^*) such that the defender's benefit $\beta_D(\delta_D, \delta_A^*)$is maximised at $\delta_D = \delta_D^*$ and the attacker's benefit $\beta_A(\delta_D^*, \delta_A)$ is maximised at $\delta_A = \delta_A^*$. To begin with, some useful notation. We denote by $opt_D(\delta_A)$ the set of values (rates of play δ_D) that optimise the benefit of the defender for a fixed rate of play δ_A of the attacker. Similarly, we denote by $opt_D(\delta_D)$ the set of values (rates of play δ_A) that optimise the benefit of the attacker for a fixed rate of play δ_D of the defender.

4.1 Determining the Piecewise Functions $opt_D(\delta_A)$

To determine $opt_D(\delta_A)$ we need to compute the derivative of $\beta_D(\delta_D, \delta_A)$ for a fixed δ_A. We consider three cases for each piece of the piecewise function of β_D.

Case A: $\delta_D \leq d$

The benefit formula for this case corresponds with the first piece of the benefit function of the defender obtained in the previous chapter:

$$\beta_D(\delta_D, \delta_A) = 1 - \frac{k_D}{\delta_D} \tag{4.1}$$

The function is of the type $1-1/x$, see figure 4.1. The root of the benefit function and the root of the first derivative are as follows:

$$\beta_D(\delta_D, \delta_A) = 0 \qquad => \qquad \delta_D = k_D \tag{4.2}$$

$$\frac{\partial \beta_D(\delta_D, \delta_A)}{\partial \delta_D} = 0 \qquad => \qquad k_D = 0 \tag{4.3}$$

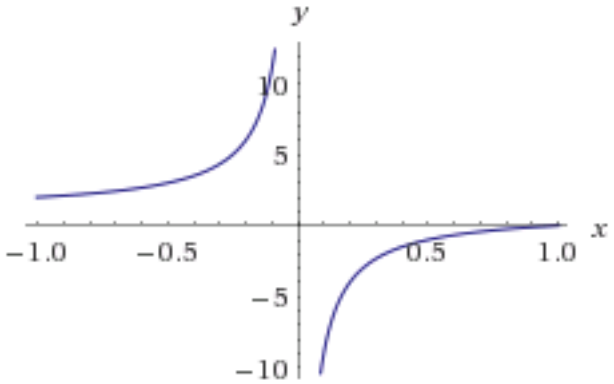


Figure 4.1: function of type $1 - 1/x$

This means that if $\delta_D < k_D$ the function is negative and the defender will therefore not play, if $\delta_D > k_D$ the function is positive. If $k_D < 0$ the function will decrease and the defender will again not play (cost cannot be negative). Assuming that costs are always non-negative $k_D > 0$, the function will increase, meaning that the slower the defender plays, the larger the benefit.

Case B: $d \leq \delta_D \leq \delta_A + d$

The benefit formula for this case corresponds with the second piece of the benefit function of the defender obtained in the previous chapter:

$$\beta_D(\delta_D, \delta_A) = 1 - \frac{\delta_D}{2\delta_A} - \frac{d^2}{2\delta_D\delta_A} + \frac{d}{\delta_A} - \frac{k_D}{\delta_D}$$

To know if the function decreases or increases we take the partial derivative of this formula for a fixed δ_A:

$$\frac{\partial \beta_D(\delta_D, \delta_A)}{\partial \delta_D} = -\frac{1}{2\delta_A} + \frac{k_D}{\delta_D^2} + \frac{d^2}{2\delta_D^2\delta_A}$$

The stationary points (maximum, minimum) can be found by setting the first derivative equal to zero and finding the roots of the resulting equation:

$$\frac{\partial \beta_D(\delta_D, \delta_A)}{\partial \delta_D} = 0 \qquad => \qquad \delta_D = \sqrt{2\delta_A k_D + d^2}$$

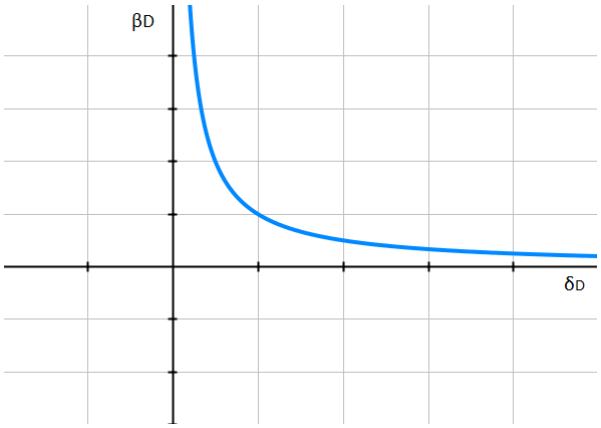


Figure 4.2: The benefit function is of the shape of 1/x and is always decreasing if $\delta_A + 2(d - k_D) > 0$.

Given the sign of the coefficient of δ_D^2, this leads to the following deduction: The function increases on $[0, \sqrt{2\delta_A k_D + d^2}]$ and is decreasing on $[\sqrt{2\delta_A k_D + d^2}, \infty]$. So we have a maximum at $\delta_D = min\{\delta_A + d, \sqrt{2\delta_A k_D + d^2}\}$ and δ_D has to be larger than d. Taking the minimum of the two values is needed because δ_D cannot be larger than $\delta_A + d$.

Case C: $\delta_D \geq d + \delta_A$

The benefit formula for this case corresponds with the third piece of the benefit function of the defender obtained in the previous chapter:

$$\beta_D(\delta_D, \delta_A) = \frac{\delta_A}{2\delta_D} + \frac{d}{\delta_D} - \frac{k_D}{\delta_D} = \frac{\delta_A + 2(d - k_D)}{2\delta_D}$$

Given that δ_D is always positive, the benefit function can be either increasing or decreasing depending on the numerator of the above fraction.

For $\delta_A + 2(d - k_D) > 0$ the benefit will always be positive but decreasing, see figure 4.2. The defender will always play as fast as he can if $\delta_A + 2(d - k_D) > 0$ for $k_D < d$ or $k_D > d$ because δ_A will be positive in either case.

For $\delta_A + 2(d - k_D) < 0$, the benefit will always be increasing but negative so the defender will not play. See figure 4.3.

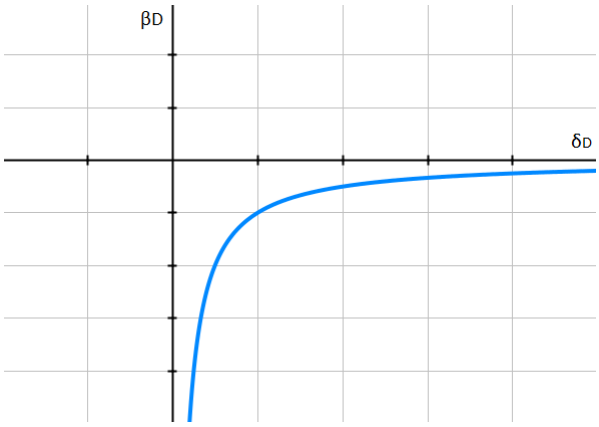


Figure 4.3: The benefit function is of the shape of -1/x and is always increasing if $\delta_A + 2(d - k_D) < 0$.

4.1.1 Best Responses of δ_D

The optimum functions are piecewise functions. We distinguish three cases for different values of δ_A depending on the values of k_D and d. Figure 4.4 shows on the left how the three cases can be combined in a single continuous function (shown as the thick line). On the right, the derivative function indicates at which points the benefit will change. It is interesting to note that the derivative is also continuous, meaning that the benefit function is smooth and has no rough edges or corners. This implies that there are no sudden changes in benefit for small changes of period. The points that are interesting to look at are the points of the derivative function that intersect with the x-axis (δ_A). These points are the roots of the derivative function and show where the benefit function is increasing or decreasing and therefore give the value of the maximum benefit of the defender.

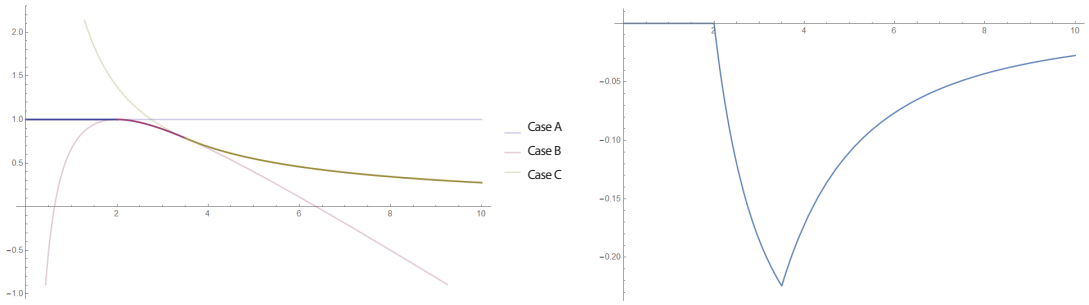


Figure 4.4: Benefit function and derivative for a cost $k_D = 0$ *and with* $d(= 2)$ *and* $\delta_A = (1.5)$ *not equal to zero.*

$\delta_A < 2(k_D - d)$

For case A, the function is increasing if $k_D > 0$. If $k_D < 0$ the function is decreasing but a positive cost is assumed. The function will therefore never decrease. $k_D = 0$ is an edge case and will be covered later in the paragraph of edge cases. So for case A we can conclude that the function is always increasing. In the next case, case B, the benefit function will increase in the interval $[0, 2(k_D - d)]$. In case C, for $\delta_A < 2(k_D - d)$, the benefit function will always increase. Case B and case C are both increasing and both functions are negative. The benefit will never become positive as such, the defender will try to not to play. The maximum benefit is achieved for $\delta = \infty$. This can be seen in (A) in figure 4.5.

$\delta_A = 2(k_D - d)$

The delay can not be a negative value. So for δ_A, a positive period, to be equal to $2(k_D - d)$, k_D has to be positive.
For case A, if $k_D > 0$ the benefit function is always increasing. In case B, the value of $\delta_A = 2(k_D - d)$ has to be filled in the root, $\sqrt{2k_D\delta_A + d^2}$. This gives the value $2k_D - d$. $2(k_D - d) < 2k_D - d$ so this means that δ_A is smaller than $\sqrt{2k_D\delta_A + d^2}$, so the function is increasing. A maximum is achieved for δ_D equal to the maximum of $\left\{\delta_A + d, \sqrt{2k_D\delta_A + d^2}\right\}$. If we fill in $\delta_A = 2(k_D - d)$, the two formulas have the same result so they are both a maximum. If we fill in the value of $\delta_A = 2(k_D - d)$ in case C, the benefit will always be 0. Case C is the case for all values $\delta_D - d \geq \delta_A$, so $\beta_D = 0$ is true for all $\delta_D \in [\delta_D - d, \infty]$. The defender's maximum benefit is thus achieved for all $\delta_D \in [\delta_D - d, \infty]$. See (B) in figure 4.5.

$\delta_A > 2(k_D - d)$

Again for case A, we assume that k_D is positive which implies that the benefit function for the defender is always increasing. If $\delta_A > 2(k_D - d)$, the benefit function of the defender in case B will only increase in the interval $]2(k_D - d), \sqrt{2k_D\delta_A + d^2}$. The maximum is achieved for δ_D equal to the maximum of $\left\{\delta_A + d, \sqrt{2k_D\delta_A + d^2}\right\}$. If we fill in $\delta_A > 2(k_D - d)$, $\sqrt{2k_D\delta_A + d^2}$ is the maximum value. In case C, for all $\delta_A > 2(k_D - d)$, the benefit function is always decreasing. If we put all the pieces together, the defender's maximum benefit is achieved for $\delta_D = \sqrt{2k_D\delta_A + d^2}$. See (C) in figure 4.5.

From this analyses we can compute $opt_D(\delta_A)$:

$$opt_D(\delta_A) = \begin{cases} \infty, & \delta_A < 2(k_D - d) \\ \left[\sqrt{2k_D\delta_A + d^2}, \infty\right], & \delta_A = 2(k_D - d) \\ \sqrt{2k_D\delta_A + d^2}, & \delta_A > 2(k_D - d) \end{cases}$$

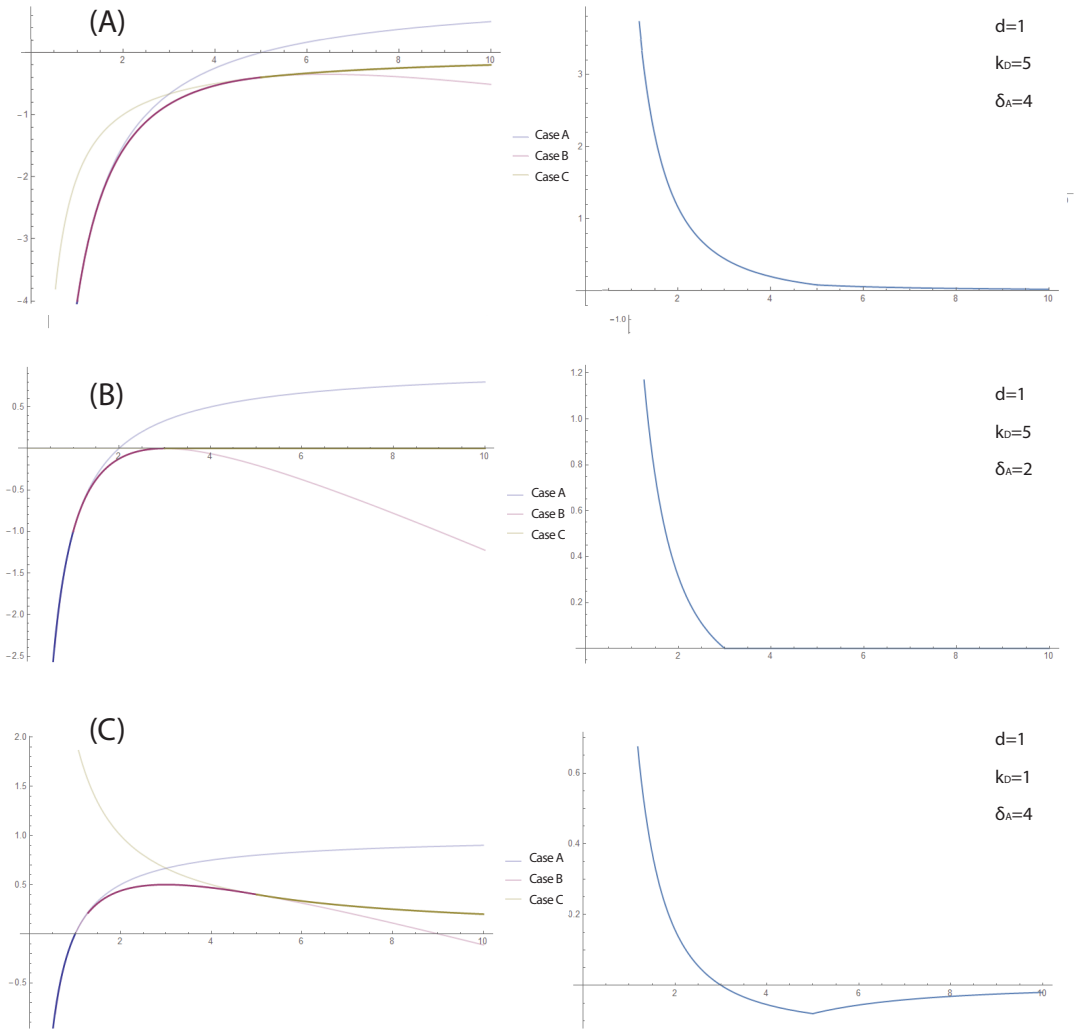


Figure 4.5: Illustration of the best response functions. The left functions are the piecewise benefit function of the defender for a certain values of k_D, d and δ_A.. The right functions are the derivatives of the benefit functions on the left. The best responses are the values where the function intersects with the x-as. (A): $\delta_A < 2(k_D - d)$, (B): $\delta_A = 2(k_D - d)$, (C): $\delta_A > 2(k_D - d)$

Edge cases:

$\delta_A = 0$ **and** $k_D = 0$

If $k_D = 0$ it means that every flip of the defender is free. So for the cost it does not matter for the defender if he plays faster or slower. $\delta_A = 0$ means that the attacker will play as fast as he can.

- if $d = 0$, if we look at $opt_D(\delta_A)$ it means that we are in the second piece. The defender will play with a rate of $[0, \infty]$.
- if $d > 0$, we have to look at the third piece of $opt_D(\delta_A)$. Here it follows that the defender has to play with a rate of d.

Combining those two leads to a rate for the defender equal to $[0, d]$. This means that the defender will never not play, which is intuitively correct if we know that the defender has no disadvantage to play because there is no cost involved.

Or more intuitively: If $\delta_A = 0$ it means that the attacker is playing as fast as possible. For the defender it does not cost anything to flip, because $k_D = 0$. We know that if $\delta_D \leq d$ the defender always has a benefit of 1. This means that for this case the defender will play with a rate $\delta_D \in [0, d]$.

$\delta_A = 0$

If $\delta_A = 0$ it means that the attacker is playing as fast as possible. The defender has still a cost for every flip. This means we have to look at different values for k_D and d.

- if $k_D > d$, it follows that the defender will not play (rate equal to ∞). This is already the case for the first piece in the piecewise function. δ_A will always be smaller than $2(k_D - d)$ if $k_D > d$.
- if $k_D = d$, it corresponds to the second piece: the defender will play with a rate of $[d, \infty]$.
- if $k_D < 0$, this corresponds with the last piece, where the rate of the defender is equal to d.

By combining the last two pieces we have d for $\delta_A = 0$ and $k_D \leq d$.

$k_D = 0$

The cost of flipping for the defender is equal to zero. If we look at the benefit function of the defender and it's derivative in figure 4.4, we can see that the benefit for case B increases until d and that the benefit in case A stays the same until d. After point d the benefit decreases in all the cases. So the period of the defender will be equal to $\delta_D \in [0, d]$.

4.1.2 Conclusion

Including the edge cases, $opt_D(\delta_A)$ is as follows:

$$opt_D(\delta_A) = \begin{cases} [0, d] & \delta_A = 0 \ \ \& \ \ k_D = 0 \\ [0, d] & k_D = 0 \\ d & \delta_A = 0 \ \ \& k_D \leq d \\ \infty, & \delta_A < 2(k_D - d) \\ \left[\sqrt{2k_D\delta_A + d^2}, \infty\right[, & \delta_A = 2(k_D - d) \\ \sqrt{2k_D\delta_A + d^2}, & \delta_A > 2(k_D - d) \end{cases}$$

4.2 Determining the Piecewise Functions $opt_A(\delta_D)$

To determine the optimal strategy of the attacker we also need to determine $opt_A(\delta_D)$ by computing the derivative of $\beta_A(\delta_D, \delta_A)$ for a fixed δ_D. We consider the three cases of the piecewise function of β_A:

Case A: $\delta_D \leq d$

The benefit formula for this case corresponds with the first piece of the benefit function of the attacker obtained in the previous chapter:

$$\beta_A(\delta_D, \delta_A) = -\frac{k_A}{\delta_A} \tag{4.4}$$

The function is of the type $-1/x$, see figure 4.6. The root of the benefit function is as follows:

$$\beta_A(\delta_D, \delta_A) = 0 \qquad => \qquad k_D = 0 \tag{4.5}$$

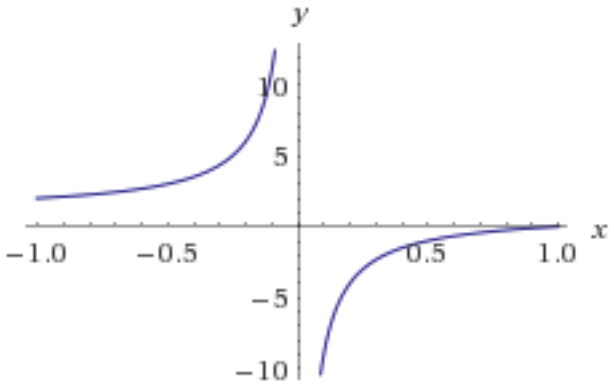


Figure 4.6: function of type $-1/x$

This means that the optimal benefit is achieved for a value $\delta_A = 0$.

Case B: $\delta_A \leq \delta_D - d$

The benefit formula for this case corresponds with the second piece of the benefit function of the attacker obtained in the previous chapter:

$$\beta_A(\delta_D, \delta_A) = 1 - \frac{\delta_A}{2\delta_D} - \frac{k_A}{\delta_A} - \frac{d}{\delta_D}$$

The derivative for a fixed δ_D is as follows:

$$\frac{\partial \beta_A(\delta_D, \delta_A)}{\partial \delta_A} = \frac{-1}{2\delta_D} + \frac{k_A}{\delta_A^2}$$

The stationary points (maximum, minimum) can be found by setting the first derivative equal to zero and finding the roots of the resulting equation:

$$\frac{\partial \beta_A(\delta_D, \delta_A)}{\partial \delta_D} = 0 \qquad => \qquad \delta_A = \sqrt{2\delta_D k_A}$$

It follows that $\beta_A(\delta_D, \cdot)$ is increasing on $[0, \sqrt{2k_A\delta_D}]$ and decreasing on $[\sqrt{2k_A\delta_D}, \infty]$ and thus has a maximum on $\delta_A = minimum\{\delta_D - d, \sqrt{2k_A\delta_D}\}$. The minimum between $\delta_D - d$ and $\sqrt{2k_A\delta_D}$ is needed because δ_A cannot exceed $\delta_D - d$ in this case.

Case C: $d \leq \delta_D - d \leq \delta_A$

The benefit formula for this case corresponds with the third piece of the benefit function of the attacker obtained in the previous chapter:

$$\beta_A(\delta_D, \delta_A) = \frac{\delta_D}{2\delta_A} - \frac{k_A}{\delta_A} + \frac{d^2}{2\delta_D\delta_A} - \frac{d}{\delta_A}$$

The derivative for a fixed δ_D is as follows:

$$\frac{\partial \beta_A(\delta_D, \delta_A)}{\partial \delta_A} = -\frac{\delta_D}{2\delta_A^2} + \frac{k_A}{\delta_A^2} - \frac{d^2}{2\delta_D\delta_A^2} + \frac{d}{\delta_A^2} = \frac{-\delta_D^2 - d^2 + 2\delta_D d + 2\delta_D k_A}{2\delta_A^2\delta_D}$$

The stationary points (maximum, minimum) can be found by setting the first derivative equal to zero and finding the roots of the resulting equation:

$$\frac{\partial \beta_A(\delta_D, \delta_A)}{\partial \delta_D} = 0 \qquad => \qquad \delta_D^2 - \delta_D(2k_A - 2d) + d^2 = 0$$

The roots of this quadratic equation are $\delta_D = d + k_A \pm \sqrt{2dk_A + k_A^2}$. Because for this case $d \leq \delta_D$, we only look at root $\delta_D = d + k_A + \sqrt{2dk_A + k_A^2}$. $\sqrt{2dk_A + k_A^2}$ is bigger than k_A so if we subtract it from $d + k_A$, we are in the part where δ_D is smaller than d.
It follows that $\beta_A(\delta_D, \delta_A)$ is increasing and non-positive if $\delta_D > d + k_A + \sqrt{2dk_A + k_A^2}$ and decreasing and positive if $\delta_D < d + k_A + \sqrt{2dk_A + k_A^2}$.

4.2.1 Best Responses of δ_A

The optimum function of the attacker is found in the same way as the optimum function of the defender. The derivatives of each piece of the piecewise benefit function of the attacker is taken. From these derivatives the best responses are derived.
Case A is an edge case and will be addressed in the paragraph about the edge cases.

$\delta_D < d + k_A + \sqrt{2dk_A + k_A^2}$

For case B, the function is increasing. The function has a maximum on $\delta_A = minimum\{\delta_D - d, \sqrt{2k_A\delta_D}\}$. For $\delta_D < d + k_A + \sqrt{2dk_A + k_A^2}$, the minimum of the two values is $\delta_D - d$.
For case C, if $\delta_D < d + k_A + \sqrt{2dk_A + k_A^2}$, the benefit function is increasing. Because we have a continuous function and the benefit is still increasing and negative in case C, the defender will try not to play. The defender's optimum benefit is achieved in $\delta_A = \infty$. See (A) in figure 4.7.

$\delta_D = d + k_A + \sqrt{2dk_A + k_A^2}$

For case B, the function is increasing. The function has a maximum on on $\delta_A = minimum\{\delta_D - d, \sqrt{2k_A\delta_D}\}$. For $\delta_D = d + k_A + \sqrt{2dk_A + k_A^2}$, the minimum of the two values is $\delta_D - d$.
For case C, $\delta_D = d + k_A + \sqrt{2dk_A + k_A^2}$ is a root so the benefit function of the attacker will be equal to zero. Because case C is the case where δ_A has to be bigger or equal to $\delta_D - d$, $\beta_A = 0$ is valid for all $\delta_A \in [\delta_D - d, \infty]$.
Putting the pieces together gives us the optimum benefit for the attacker for all $\delta_A \in [\delta_D - d, \infty]$. See (B) in figure 4.7.

$\delta_D > d + k_A + \sqrt{2dk_A + k_A^2}$

For case B, the function is increasing. The function has a maximum on $\delta_A = minimum\{\delta_D - d, \sqrt{2k_A\delta_D}\}$. For $\delta_D > d + k_A + \sqrt{2dk_A + k_A^2}$, the minimum of the two values is $\delta_D - d$.
For case C, if $\delta_D > d + k_A + \sqrt{2dk_A + k_A^2}$ the function is decreasing. So the optimum benefit for all the cases is when $\delta_D = \delta_D - d$. See (C) in figure 4.7.

From this analyses we can compute $opt_A(\delta_D)$:

$$opt_A(\delta_D) = \begin{cases} \infty & \delta_D < d + k_A + \sqrt{2dk_A + k_A^2} \\ [\delta_D - d, \infty], & \delta_D = d + k_A + \sqrt{2dk_A + k_A^2} \\ \delta_D - d, & \delta_D > d + k_A + \sqrt{2dk_A + k_A^2} \end{cases}$$

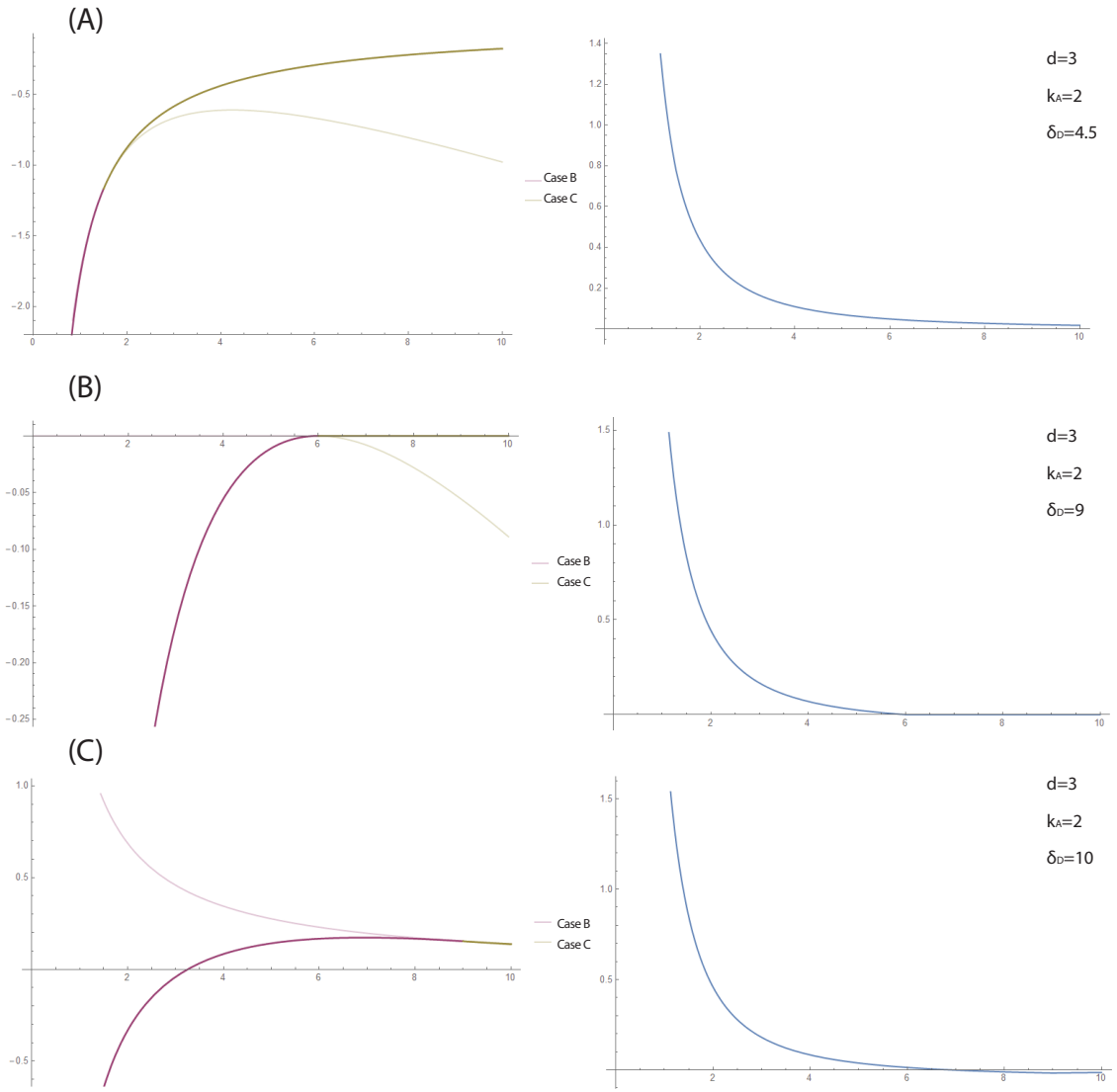


Figure 4.7: Illustration of the best response functions. The left functions are the piecewise benefit function of the attacker for a certain values of k_A, d and δ_D. The right functions are the derivatives of the benefit functions on the left. The best responses are the values where the function intersects with the x-axis. (A): $\delta_D < d + k_A + \sqrt{2dk_A + k_A^2}$, (B): $\delta_D = d + k_A + \sqrt{2dk_A + k_A^2}$, (C): $\delta_D > d + k_A + \sqrt{2dk_A + k_A^2}$

.

Edge cases:

$\delta_D = 0$ **and** $k_A = 0$

If $\delta_D = 0$, looking at the benefit formula of the attacker, it follows that the period of the defender will always be equal or smaller than the delay. The delay cannot be negative. So from case $\delta_D \leq d$ it follows that the attacker will always have a benefit equal to zero, because $k_A = 0$. For the attacker it does not matter what he plays. The benefit will always be the same. The optimal strategy in this case for the attacker will be playing at a rate $\delta_A \in [0, \infty]$.

$k_A = 0$

If the cost of playing for the attacker is zero, the attacker can play as fast as he can. If $\delta_D \leq d$, looking at the benefit formula for the attacker, it does not matter what the attacker does, his benefit will always be zero. He will play with a rate $\delta_A \in [0, \infty]$. We can merge this with the previous case where $\delta_D = 0$ and $k_A = 0$. If $\delta_D > d$, the attacker will play as fast as he can. He will play with a rate of $\delta_A = 0$.

$\delta_D = 0$

If $\delta_D = 0$ it follows from the case $\delta_D \leq d$ that the attacker will always have a negative benefit, unless the cost is zero. If the cost is zero we have again the case of $\delta_D = 0$ and $k_A = 0$. But if the cost is non-zero, the optimal strategy of the defender is not moving at all.

4.2.2 Conclusion

Including the edge cases yields the following for $opt_A(\delta_D)$:

$$opt_A(\delta_D) = \begin{cases} \infty & \delta_D = 0 \\ [0, \infty] & \delta_D \leq d \ \ \& \ \ k_A = 0 \\ 0 & \delta_D > d \ \ \& \ \ k_A = 0 \\ \infty & \delta_D < d + k_A + \sqrt{2dk_A + k_A^2} \\ [\delta_D - d, \infty], & \delta_D = d + k_A + \sqrt{2dk_A + k_A^2} \\ \delta_D - d, & \delta_D > d + k_A + \sqrt{2dk_A + k_A^2} \end{cases}$$

4.3 Conclusion

The piecewise functions opt_D and opt_A give some interesting results.

- When the defender plays faster than the delay, the attacker will either have a negative benefit or a benefit equal to zero if the flips do not cost anything. It

is for the defender a target to be able to play at a rate smaller or equal to the delay.

- If the defender can play with a cost equal to zero, the attacker will not play. The defender can play at any rate he wants, and the attacker is always disadvantaged by his delay.
- If the cost of the defender is non-zero, then, depending on the ratio between it's cost and the delay, from a certain value for the speed of the attacker, the defender will not play. The same thing goes for the attacker. This result is similar to the results obtained for the original FlipIt game.

Chapter 5

Models for the Delay

The formalization of the FlipIt game with a delay relies on some value d that represents the time needed to infect a sufficient number of nodes in a network after the initial infection. This chapter provides the reader with more insight on how to calculate the value of this parameter d.

The spreading of malware has already been extensively researched. Because of the many different types of propagation methods it is hard to define a single model that can model all of them. Modelling the spread of malware depends on two key factors: the method used for the propagation and the graph of the network in which the malware will spread.
Viruses and worms are the two most researched types of malware. Since the spreading of viruses requires human interaction, their propagation delay depends on (hard to predict) human behaviour. Worms on the other hand, spread without human intervention, and their propagation is therefore easier to model. Since the purpose of this chapter is only to illustrate how a delay can be calculated, we will limit this chapter to propagation models of worms.

The overview of this chapter will be as follows: Section 5.1 presents an overview of the most frequently used propagation methods by worms. These propagation methods will be covered by different models in section 5.2. A set of models is selected to illustrate the different possible ways to extract parameter d. Section 5.3 introduces an easy method to calculate the propagation delay of a worm. In the last section 5.4, a method based on the PageRank algorithm of Google is used to determine the probability of infection after a certain time period.

5.1 Methods of Propagation

In the context of malware propagation, there are two kinds of APTs. First, there are APTs that launch an attack on just one target node of a network. The mechanism these APTs use to propagate malware is the dropper mechanism. The dropper is the initial attack vector that compromises the single targeted node of the system.

The second kind of APTs target multiple nodes. These APTs also use a dropper mechanism but have an additional mechanism for self-propagation in order to propagate themselves to the multiple targeted nodes of the system. If the APT uses virus spreading, the virus infects one node on the network and has to wait for human interaction to spread. As such the spreading speed depends on the human interaction, and modelling virus propagation therefore requires modelling human behaviour. If an APT uses a worm propagation method it will spread by itself after it has been dropped on the network. Given the additional complexity of incorporating the human factor in a spreading method and that this chapter is merely meant as illustration on calculating the delay for use in the game theoretic approach, this chapter will be limited to worm propagation models for APTs.

Infection by worms start by dropping the worm on the network. Different dropping mechanisms can be used, whereby the initial attack can be random or targeted at a specific node. Frequently used mechanisms are USB sticks (given to a specific person or left behind to be picked up by a random person), email, or malicious software through phishing or trojan horses. To determine the total delay, however, the propagation strategy is of higher importance than the drop mechanism. The propagation strategy is used to determine the next nodes the worm will spread to. The following are common propagation methods:

Selective Random scanning: The worm randomly selects a part of the selected IP address space instead of scanning the whole address space. The selected IP address space is a certain IP address area that the attackers are planning to attack. The reserved address blocks and the unassigned addresses are excluded from the address space. The rate of success for randomly chosen IP addresses is very low, but it is easy to implement. Example of such worms are *Code Red* [21] and *Slammer* [13].

Localized scanning: A worm that uses localized scanning will scan for hosts in the local address space. This method is used by the *Code Red II* [21] and *Nimda worm* [21].

Sequential scanning: With sequential scanning, the worm will scan the IP addresses sequentially. This means that once a vulnerable host is compromised, it will look for IP addresses that are near to this host. For example, the address of the host is A, the next addresses that the worm will scan are A+1, or A-1. This method is used by the *Blaster worm* [30].

DNS random scanning: Another strategy is a kind of strategy in which the DNS infrastructure is used to locate a new target address. The IP address table from a DNS server is acquired from DNS records. The speed of a DNS scanning worm in the IPv6 internet is comparable to the speed of an IPv4 random scanning worm. The IP addresses stored in the address table are only hosts with public domain names. This propagation method is used by *MyDoom* [4].

Routable scanning: The worms using routable scanning acquire target IP addresses based on the routing information in a network. Through the BGP routing tables they can scan the routable address space. This method is three times faster than a traditional worm that uses random scanning. Examples are *Spyb0t* or *network Bluepill*.

Topology-based Worms: Email and other client application worms: an email worms uses the email systems to find email addresses to propagate. Other client applications can include: Internet Relay Chat (IRC), Instant Messenger (IM), and a variety of peer-to-peer file sharing systems, which have been used by worms to propagate in a similar way as email worms. For example, the *Kak worm* [18] is a JavaScript computer worm that spread itself by exploiting a bug in Outlook Express.

Modelling the propagation methods described above can help us to find the time needed for a worm to infect the whole network. This will be equal to the d parameter in the FlipIt model with propagation delay.

5.2 Models for Worm Propagation

Early work on worm propagation models are based on the spread of real-world epidemic diseases. These models are based on the transition state of each node in the network.

A node in a network can be in three different states: susceptible, infected or removed. A susceptible node is a node that is vulnerable to infection. An infected node is a node that has been compromised and can infect other nodes. A removed node is dead or immune, which means it cannot be infected again by worms. With these three states three main propagation models are proposed: SI (Suspected-Infected), SIR (Suspected-Infected-Removed) and SIS (Suspected-Infected-Suspected). In the SI model, a node that has once been infected, stays infected. In the SIR model, an infected node can be removed afterwards. This node cannot become infected again. In the SIS model a node can become susceptible again after it has been infected. Currently, various other models have been proposed based on these three models (Wang et al. [26], Qing and Wen et al. [16], Xiang t al. [28] and Serazzi et al. [19]). We are only interested in models that will allow us to find the delay.

The most recent survey on common propagation methods is the one by Wang et al. [26]. It encompasses the results of older surveys and explicitly focuses on propagation methods, while other papers also focus on other aspects such as detection mechanisms and containment systems. Table 5.1 provides the taxonomy on worm modelling, given in [26] (only analytic modelling methods). All models that are based on the SI model type are appropriate, because in our model every node that is infected will stay infected. The other propagation models, SIR and SIS, are only

Worm Propagation Models	Network Topology	Graphical Representation of Topology	Propagation Process	Model Type
Classical Simple Epidemic Model	H	UG	C	SI
Uniform Scan Worm Model	H	UG	C	SI
RCS Model	H	UG	C	SI
Classical General Epidemic Model	H	UG	C	SIR
Two-Factor Model	H	UG	C	SIR
AAWP Model	H	UG	D	SIR
Bluetooth Worm Model	H	UG	D	SI
Local Preference Model	Non-H	UG	C	SI
LAAWP Model	Non-H	UG	D	SIR
Logic 1-0 Matrix Model	R/PL	DG	D	SIR
Spatial-temporal Model	H/PL	DG	D	SIS

H: homogeneous mixing; R: random network; PL: power-law network;UG: undirected graph; DG: directed graph; C: continuous-time event; D: discrete-time event; SI: susceptible-infected; SIR: susceptible-infected-removed; SIS: susceptible-infected-susceptible

Figure 5.1: Taxonomy of worm modelling.Table based on only analytic worm propagation models given in [26]. The models in green are the models illustrated in this paper.

interesting if there is a possibility to reduce the models to an SI model. The following section will extract the delay parameter from some of the models. These models are not meant as an exhaustive list, but rather as an illustration to the possible ways of extracting the delay out of an propagation model.

5.2.1 Simple Epidemic Model

A Simple epidemic model is another name for the general SI model. This model assumes that each node in the network can be either susceptible or infected. Once a node is infected it will stay infected. Every node in the network has the same chance to be infected. The simple epidemic model is considered to be of a fixed size, meaning that no nodes are added to the network or removed. The model for a fixed population is as follows:

$$\frac{dI(t)}{dt} = \beta I(t)[N - I(t)] \tag{5.1}$$

where *I(t)* is the number of infected nodes at time t, β is the propagation rate, and N is the number of nodes in the network. In the beginning, $t = 0$, *I(0)* nodes are infected. All the other nodes, $N - I(0)$, in the network are susceptible. The solution

of this equation is the following logistic curve:

$$I = \frac{e^{\beta(t-T)}}{1 + e^{\beta(t-T)}} \tag{5.2}$$

where T is a time parameter representing the point of maximum increase in the growth.

To extract the delay from the formula we need the total number of nodes in the network. If I in formula 5.2 equals the total number of nodes, variable t will be the value of the delay that we need. A limitation of this model is that it is only applicable to nodes in a homogeneous network. A homogeneous network is a network where every node has approximately the same degree, the number of connections the node has. This model is thus suitable for propagation of worms that are topology independent (e.g scan-based worms), because every node has the same chance to be infected by another node in the network. The *Code Red* worm, which is a random scan based worm, has been analysed by this kind of model in [21]. Another limitation is that the graph is presented as an undirected graph. This means that the spread can always happen in both ways, which may not be suitable for every worm propagation.

5.2.2 RCS model

The RCS model stands for Random Constant Spread model and was developed by Paxson and Weaver at Stanford [21]. It is a model derived from the classical Simple Epidemic Model and was used to analyse the *Code Red I* worm. For this model it is assumed that the worm owns a good random number generator that is properly seeded.

Let N be the total of vulnerable hosts in the network that can be potentially infected. It is assumed that no system is patched, shut down, deployed or disconnected. This means that the number of hosts in the system stays constant. Let K be the initial compromise rate. This is the rate per hour at which the worm can find and compromise hosts at the beginning of the infection. K is a global constant and therefore does not depend on the speed of the network or the processor speed. Every machine can infect only one other machine after it has been compromised. The rate that a worm can find hosts can not be increased. Let T be the moment of the start of the infection. Variable a is the proportion of vulnerable hosts that has been compromised. Variable t is the time in hours.

The formula to model the spread of the worm is as follows:

$$Nda = (Na)K(1 - a)dt \tag{5.3}$$

It tells how many vulnerable machines will be compromised in the next amount of time dt, when the proportion of the machines a that are already compromised is known.

From this, it follows the simple differential equation:

$$\frac{da}{dt} = Ka(1-a) \tag{5.4}$$

With the following solution:

$$a = \frac{e^{K(t-T)}}{1+e^{K(t-T)}} \tag{5.5}$$

To extract the delay from this formula, we need to know the maximum numbers of hosts that can be infected before the total system is compromised. The initial compromise rate has to be approximated. t is the time variable. If a passes the proportion of nodes that have to be compromised before the whole system is compromised by the attacker, value t is the value of d in the FlipIt game with propagation delay.

A drawback of this model is that the network topology is assumed to be homogeneous and that the paths between the nodes are undirected.

5.2.3 Bluetooth worm model

The Bluetooth worm model was introduced by Yan and Eidenbenz [29]. The model captures the behaviour of the propagation of a worm that spreads through the Bluetooth protocol. A Bluetooth device that is compromised can only infect neighbour devices that are in its radio range. Let *i(t)* be the average density of infected hosts in the network at time t_0. Then:

$$i(t_{k+1}) = i(t_k) \cdot \frac{\rho(t_k)}{i'(t_k) + (\rho(t_k) - i'(t_k))e^{-\alpha' \cdot \rho(t_k)/(\rho(t_k)-i'(t_k))}} \tag{5.6}$$

where ρ(t) and β(t) are the average device density and the pairwise infection rate at time t respectively. The number of new infections out of the infection cycle is denoted by α(t). α' is used for a better estimation of the worm propagation, because the growth rate of the worm can change and this can maybe result in an overestimation of the number of new onfections. α' is determined as follows:

$$\alpha' = \frac{\rho(t_k) - t(t_k)}{\rho(t_k)} \cdot \alpha(t_k) + \frac{i(t_k)}{\rho(t_k)} \cdot \alpha(t_x) \tag{5.7}$$

To apply this model on FlipIt we again set up a threshold value for i(t) and see how long it takes to compromise this amount of hosts. The paper by Yan and Eidenbenz concluded in their work that after setting model parameters accordingly ($\lambda_{ne} = 0.2108$ the average node degree and $J_{in} = 0.2372$ the average meeting rate of neighbours), the model predicts that the time it would take to infect 99% of the devices is slightly less than one hour. So in this case the delay is equal to an hour and the amount of hosts that has to be infected by the attacker before the network is compromised is 99% of the hosts in the network. The Bluetooth worm spreads

rapidly once the density of the infected hosts reach 10 percent. The model shows that the spread in the beginning is very slowly at en early stage.

The limitation of this model is that it assumes that all the hosts in the network are homogeneously mixed and that the time increases in a discrete fashion. The graphical representation is also a undirected graph.

5.2.4 AAWP model

AAWP stands for Analytic Active Worm Propagation. The model was proposed in [2] by Chen et al. to model the discrete behaviour of a worm. The AAWP model is a SIR model which uses a death and patch rate to calculate the amount of hosts that become Removed. Yet this model can still be useful to calculate the delay because if the death and patch rate are removed, this model becomes an SI model. It is different from other SI models because it includes the time that it takes to infect a host. A host cannot infect another hosts before it is completely compromised. The model also considers the fact that a host can be scanned and hit by multiple worms at the same time.

The spread of the AAWP model with death and patch rate is characterized as follows:

$$I_{t+1} = (1 - d - p)I_t + [(1 - p)^t)N - I_t][1 - (1 - \frac{1}{\Omega})^{sI_t}] \tag{5.8}$$

where d is the death rate, p is the patch rate, I_t is the amount of infected hosts at time t, N is the number of vulnerable hosts, s is the scanning rate of the worm and Ω is the scanning space.

The spread of the worm without the death and the patch rate is as follows:

$$I_{t+1} = I_t + (N - I_t)[1 - (1 - \frac{1}{\Omega})^{sI_t}] \tag{5.9}$$

The iteration procedure with death and patch rate will stop when all the nodes in the network are infected or when the number of infected nodes remains the same. When the death and patch rate are removed from the formula the iteration procedure will stop when all the nodes are infected. So when the iteration procedure stops the delay will be equal to *t+1*.

The biggest limitation of the model is that is expressed in discrete time. Continuous models are more appropriate for large scale models. The model also uses a complete undirected graph.

5.3 Matrix based Worm Model

As can be seen in table 5.1 all propagation models depend on a specific network topology. A distinction is made between four kinds of network topologies: homogeneous, non-homogeneous, random network or power-law network. A homogeneous network is a network where every node has about the same degree of connectivity

and each connection has the same probability. A non-homogeneous network is a network in which not all the nodes have the same degree and the connections have a different probability. A random network is a topology in which each connection is chosen at random with equal probability. In a power law network the degrees of each node in the network follow the power law. Some of the nodes have a small degree of connectivity, others have a very large degree of connectivity.

However, for some of the propagation methods, the actual graph of the network matters. Each propagation method depends on having the right topology. Email worms need a topology that correspond to a social network, BGP routing worms need a topology on IP-network level. Ideally, the propagation delay should be calculated without assuming that the network has one of these predefined topologies, but rather using the actual topology of the network at hand.

This chapter proposes a method to calculate the spread of a worm in a way that allows easy integration of the network topology. This method approximate how fast a worm can infect a network. It is capable of calculating the delay for any network topology.

Proposal of a matrix based model

A computer network can be modelled by an undirected or directed graph $G =< N, E >$ where $|N|$ denotes the number of nodes in the network and $|E|$ the number of connections. This graph can be converted to an $|N| \times |N|$ adjacency matrix in which the entries represent the connections between the nodes of the network.
The matrix has a non-zero entry a_{ij} if there is a connection from node N_i to N_j.

Adjacency matrices have many interesting applications, amongst which calculating the paths between vertices: *"If A is the adjacency matrix of the directed or undirected graph G, then the matrix A^n (i.e., the matrix product of n copies of A) has an interesting interpretation: the entry in row i and column j gives the number of (directed or undirected) walks of length n from vertex i to vertex j. If n is the smallest nonnegative integer, such that for all i ,j , the (i,j)-entry of $A^n > 0$, then n is the distance between vertex i and vertex j."* source: [27] .
Using this property of an adjacency matrix, it is possible to calculate the time it takes for a worm, starting on a specific node, to infect a sufficient amount of other nodes in the network. Every matrix A^n has a non-zero ij-entry , if the worm starting in node i can reach node j in n time steps. If we sum up all the matrices $A^1 + A^2 + \ ... \ + A^{n-1} + A^n$, every i-row indicates which node j is infected by node i after time $t = n$.

Assuming a network like in figure 5.2, the corresponding adjacency matrix is the matrix A:

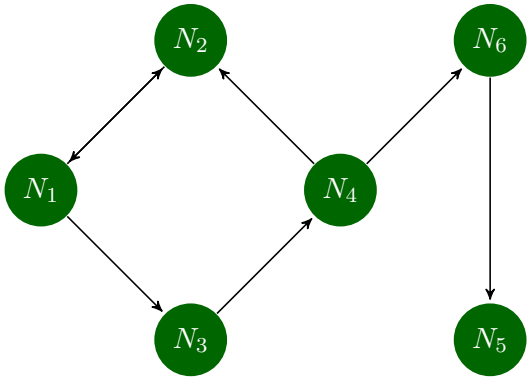


Figure 5.2: Network with 6 nodes. The arrows represent the connections between the nodes.

$$A = \begin{array}{c} \\ N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{array} \begin{array}{c} \begin{array}{cccccc} N_1 & N_2 & N_3 & N_4 & N_5 & N_6 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \tag{5.10}$$

In matrix $A \times A = A^2$, each entry represents the number of paths with length 2 from N_i to N_j:

$$A \times A = A^2 = \begin{array}{c} \\ N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{array} \begin{array}{c} \begin{array}{cccccc} N_1 & N_2 & N_3 & N_4 & N_5 & N_6 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.11}$$

Likewise, in matrix $A^2 \times A = A^3$ each entry represents the number of paths with 3 steps from N_i to N_j.

$$A \times A \times A = A^3 = \begin{array}{c} \\ N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{array} \begin{array}{c} \begin{array}{cccccc} N_1 & N_2 & N_3 & N_4 & N_5 & N_6 \end{array} \\ \begin{pmatrix} 0 & 2 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.12}$$

So, in A^n every a_{ij} entry gives the number of paths with n steps from N_i to N_j.

Calculating the sum of the three matrices $(A + A^2 + A^3)$ results in a matrix where the non-zero entries a_{ij} indicate which nodes are infected after 3 time steps if a virus is dropped on noded i:

$$A + A^2 + A^3 = \sum A^n = \begin{array}{c} \\ N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{array} \begin{array}{c} \begin{array}{cccccc} N_1 & N_2 & N_3 & N_4 & N_5 & N_6 \end{array} \\ \begin{pmatrix} 1 & 3 & 2 & 1 & 0 & 1 \\ 2 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 2 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \tag{5.13}$$

For this sample network, the matrix in 5.12 indicates that by dropping a virus on node N_1, all but node N_5 will be infected after 3 time steps. If the virus is dropped on node N_5, the virus will not spread.

With the formula $\sum A^n$ the average delay can be calculated by counting the expected delay of every node when a worm has been dropped and devide it by the number of nodes in the network.

Matrix A can also represent a graph where every link has a weight. The weight is equal to the probability that the worm spreads through that link. The outcome of $\sum A^n$ is then equal to the probability that each node has been infected after n time steps. The average delay can be calculated by setting a threshold value for the probability that a node is infected.

Conclusion

The advantage of this matrix worm model in comparison with the above models is that matrix A can represent any network topology, directed or undirected.
Using this method it is also possible to determine the shortest path through al the nodes in the network. The shortest path that is necessary to infect all nodes can be found by summing all the matrices. The first row that only has non-zero entries determines this path. From node i to the last ij-entry that became a non-zero value. Consequently, if a worm is dropped on node i, this worm will compromise the network in the shortest time possible. A defender can use this knowledge to adapt its network configurations.
These matrix calculations may seem very time consuming, but in the domain of network analytics efficient algorithms are available that are capable of calculating network attributes per node in networks of millions of nodes. This paper only provides a proof of concept of how to apply these matrices to calculate the delay.

Developing an efficient algorithm is considered as a potential avenue for further research.

5.4 PageRank Algorithm

The above method gives the propagation delay with the infection starting on a certain node. The following method, the PageRank algorithm, will give the probability that a node is infected, with the assumption that every node in the network is equally likely to be infected as a starting point.

The PageRank algorithm was introduced by Page and Brin in 1998 as one of the main features of the search engine Google to improve the search results. PageRank models the human behaviour when users surf through the net. It can also model *random surfers*. A random surfer represents the probability that a surfer gets bored and randomly visits another page which is not linked with the initial page. The probability that he will visit a random page is the PageRank of that page. A page will have a high PageRank if many pages point to this page. This means that the page is well-cited through other pages and may be important to look at. The main idea of the PageRank algorithm is to look at the number of (important) web pages that point to a particular page. The ranking of this page then depends on the number of outgoing links and the importance of the pages that link to this page. With a probability of P, the surfer will follow a link of the page to another page, and with a probability of $1 - P$ the surfer will surf to a random page. The PageRank algorithm is expressed as follows:

$$PageRank(A) = P \sum_{i \in N_A} \frac{PageRank(i)}{d_{out,i}} + (1 - P) \cdot e_A \tag{5.14}$$

where N_A is the set of pages, $PageRank(i)$ is the page rank of web page i, $D_{out,i}$ is the number of outgoing links of page i, $(1 - P)$ the probability that a random page is searched, and e_A the restart value for web page A which is often uniformly distributed among all web pages.

The PageRank of each page on the internet is the dominant eigenvalue of the Google Matrix. The Google Matrix is a stochastic matrix and represents a graph where every edge denotes a link between pages. Through the power iteration [23], the PageRank of all the pages can be calculated iteratively.

The Google Matrix of a graph is defined as follows:

$$M = (1 - p) \cdot A + p \cdot B \quad where \quad B = \frac{1}{n} \cdot \begin{bmatrix} 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{5.15}$$

Matrix A corresponds to an $n \times n$ matrix (n defines the number of pages) that represents the graph where every entry corresponds to a non-zero entry if there is a

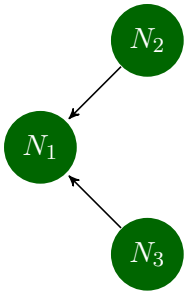


Figure 5.3: A representation of a graph with three nodes. Node 1 is a dangling node with no outgoing links.

link from one page to the other page. p defines the damping factor and indicates the probability that a surfer quits a current page and continuous on a new random page. Because the surfer can go to any random page, each page has a probability of $\frac{1}{n}$ to be chosen. This corresponds to matrix B.

The PageRank vector of a graph, with the transposition matrix A and the damping variable p, is equal to the probabilistic eigenvector of the matrix M, corresponding to the eigenvalue 1. This can be calculated by the use of the power iteration.

We can use the Google matrix to calculate the probability that a node in a graph is infected by a worm after a certain amount of time. When a worm uses the topology of the graph to propagate the damping variable p must be set equal to zero, as in the worm will not suddenly attack a node that is not connected. If the propagation of a worm is topology independent, the damping variable p can be set to 1.
The ranking of nodes provides interesting avenues for further research for the case where the defender flips a subset of nodes (rather than the entire network). It can be advantageous to flip the nodes with a higher PageRank more often than the other nodes. A higher PageRank means that the node has a higher probability of being compromised by a worm.

Chapter 6

Conclusion

This paper presented an adaptation to the basic FlipIt game by [24] to model an attacker with a delay.

6.1 General Results and Conclusions

Our FlipIt model with worm propagation delay showed us some interesting results.

- When the defender plays faster than the delay, the attacker will either have a negative benefit or a benefit equal to zero if the flips do not cost anything. It is for the defender a target to be able to play at a rate smaller or equal to the delay.
- If the defender can play with a cost equal to zero, the attacker will not play. The defender can play at any rate he wants, and the attacker is always disadvantaged by his delay.
- If the cost of the defender is non-zero, then, depending on the ratio between it's cost and the delay, from a certain value for the speed of the attacker, the defender will not play. The same thing goes for the attacker. This result is similar to the results obtained for the original FlipIt game.
- The optimum strategy for the defender and the attacker is depending on the value of d. In the beginning a non-adaptive strategy was assumed for both players, so no player gets feedback in the game. None of the players have knowledge about the real value of d, so both players have to make a guess. By using the model matrix based model, different kind of attacks can be simulated and an average delay can be used.

While the first two results are intuitively clear, in the third case, the mathematical analysis is important to determine the exact ratio's at which it is no longer beneficial for the defender to play.
In the next section we list a set of extensions that might be interesting for further research.

6.2 Further Work

Nash equilibria
This paper concluded with the best responds functions of the attacker and the defender. The next step would be to calculate the Nash equilibria. For these calculation multiple cases have to be examined, i.e. all the possible cases regarding the mutual relations between k_A, k_D and d.

Analysing other renewal strategies
By analysing other renewal strategies, it might be interesting to find out if the original result regarding periodic versus other renewal strategies of the basic FlipIt game still stands. This result implied that periodic strategies strongly dominate the other renewal strategies if an opponent plays with a non-adaptive strategy.

Delay for the defender
This paper only assumed that the attacker had a delay. It could be interesting to have a defender with a delay. This situation could happen if it takes some time to make a patch to a system or when a new exploit is found.

Variable delay
This paper assumed that the propagation delay has a fixed value. This can be relaxed by giving the attacker a delay could vary. The delay always negatively impacts the benefit of the attacker, so the attacker will always choose the lowest delay. But if the attacker always sends a new worm at every flip, it could be that the delays vary. In that regard, a variable delay would be more accurate to simulate a real world case.

The defender flips a subset of nodes
Our model assumes that if the defender flips, he flips the whole network. An option for the defender would be to flip a subset of nodes in the network. If the costs of flipping are related to the amount of nodes that are flipped, this option might be interesting to look at. It could be that a certain subset of nodes is found that protects the whole network. The PageRank matrix explained in Chapter 5 can help finding the right nodes.

As of today APTs become more and more extraordinary pieces of malicious code. There are APTs known to survive military-grade disk wiping and reformatting, and even after reformatting and reinstalling the operating system they are able to send sensitive data (Equation group [22]). This means that even if you know that an APT is on your network, the actual practical “flip” has to exist and be known by the defender.
Most often the best way to secure a system is to find the weakest link. This is mostly the employee. An effective way to counter this problem is to raise awareness about risks of infections amongst the employees. Even in the presence of good security

protection layers, spam filters, and firewalls, the opening of a phishing mail or use of an unknown USB stick by one of your employees may be sufficient to infect a PC on the network, from which the virus can spread further. A system can however never be 100% waterproof. If an infection has occurred and the defender knows a countermeasure, the FlipIt game will help to determine the most effective pace at which countermeasures need to be taken.

Appendix A

The First Appendix

Flip the virus: A gametheoretic approach to cybersecurity

Sophie Marien
DistriNet KULeuven
sophie.m.marien@gmail.com

***Abstract*—Recently, high profile targeted attacks such as the attack on Belgacom (a major Belgian telcom), have demonstrated that even the most secure companies can still be compromised, and that moreover such attacks can go undetected for a while. FlipIt has been proposed by a group of researchers at RSA to model such stealthy takeovers. It is a 2-players game composed of a single attacker, a single defender and a single shared resource. The players will try to gain control over the shared resource and they do this in a stealthy way. FlipIt does however not take into account that a move may not be instantaneous, but has a certain delay. In this paper we adapt FlipIt such that we can use it to model the game of defending a company network that is attacked by a virus. The FlipIt formulas are adapted such as to take the delay for virus propagation into account.**

I. Introduction

IN THIS ERA where digitalization becomes prominent in every aspect of our lives, where technology is growing fast and where businesses are always under attack, security becomes an issue of increasing complexity. Without security, there is no protection to keep somebody out of a system. It is the same as leaving the door of your house wide open for everyone to come in.

Why is it so important to keep a system secure? Many businesses store confidential information on clients, which can be lost and possible be abused by competitors through data leakage. Also, disruption caused by DDoS attacks, may results in businesses failing to meet their service-level agreements. Ultimately, system and network security helps protecting a business's reputation, which is one of its most important assets.

A particular kind of frequently occurring threats are Advanced Persistent Threats (APT). An APT is a targeted cyber attack that targets organisations in a stealthy way and that can stay undetected for a long period. This makes it so hard to protect a network or a system against an APT. Bruce Schneier describes an APT as something different and stronger than a conventional threat: "*A conventional hacker or criminal isn't interested in any particular target. He wants a thousand credit card numbers for fraud, or to break into an account and turn it into a zombie, or whatever. Security against this sort of attacker is relative; as long as you're more secure than almost everyone else, the attackers will go after other people, not you. An APT is different; it's an attacker who - for whatever reason - wants to attack you. Against this sort of attacker, the absolute level of your security is what's important. It doesn't matter how secure you are compared to your peers; all that matters is whether you're secure enough to keep him out*" - Bruce Schneier [1].

Since it is so difficult to protect a system or a network against APT's, researchers have been looking for effective ways to predict in advance which defence strategy might be the better one. Game theory is gaining increasing interest as an effective technique to model and study Cyber Security. Game theory analyses the security problem as a game where the players are an attacker and a defender of a system, and where both players have to make decisions. In particular, both players will aim for the strategy that results in a maximal benefit for them. Researchers at RSA made a game theoretic framework to model targeted attacks. They study the specific scenario where a system or network is repeatedly taken over completely by an attacker and this attack is not immediately detected by the defender of the system or network. In game theory, such a game is known as "FlipIt" [2]. This is a two players game where the attacker and the defender are competing to get control over a shared resource. Both players do not know who is currently in control of the resource until they move. In FlipIt every move gives them immediately control over the resource. But what if the attacker moves and it takes a while before the attacker gets full control over the resource? FlipIt does not take into account that a move may not be instantaneous, but has a certain delay. Consider for example a network with different nodes (laptops, datacenters) as a resource. The attacker drops a virus on one of the nodes and then wait till this virus infects the whole network. The attacker will only be in control of the resource once the whole network is infected.

This paper proposes the adaptation of the FlipIt formulas as presented in [2] such as to take the delay for virus propagation into account. In the next section we first present the original FlipIt game. Then section III presents the FlipIt game with virus propagation. Section IV presents some related work. Section V concludes the paper and presents avenues for further research.

II. The FlipIt game

FlipIt is a game introduced by van Dijk et al. To understand how to model a FlipIt game with virus propagation it is important to get familiar with the concepts of the normal FlipIt game and its notations. Therefore, we first explain

the framework of FlipIt and introduce the most important formulas that will be used throughout the paper.

FlipIt is a two-players game with a shared single resource that the players want to control as long as possible. The shared resource can be a password, a network or a secret key depending on the setting being modelled. In the remainder of the paper we name the two players the attacker, denoted by the subscript *A* and the defender, denoted by subscript *D*.

The game begins at $t = 0$ and continues indefinitely ($t \to \infty$). The time in the game is assumed as being continuous, but a discrete time could also be considered. To get control over the resource, the players i, with $i \in \{A, D\}$, can flip the resource at any given time. A flip will be regarded as a move from a player *i*. Each move will imply a certain cost k_i and the cost can vary for each player. Both players will try to minimize their cost. Adding a cost will prevent players to move too frequently.

The unique feature of FlipIt is that every move will happen in a stealthy way, meaning that the player has no clue that the other player (his adversary) has flipped the resource. For instance, the defender will not find out if the resource has already been compromised by the attacker until he flips the resource himself. The goal of the player is to maximize the time that he or she has control over the resource while minimizing the total cost of the moves. A move can also result in a "wasted move", called a flop. It may happen that the resource was already under control by the player. If the player moves when he or she has already control over the resource, he or she would have wasted a move since it does not result in a change of ownership, so the cost is wasted.

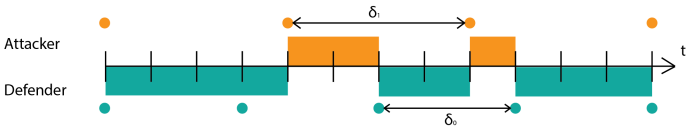


Fig. 1: A representation of a FlipIt game where both players are playing periodically and at discrete time intervals. Every move or flip is indicated by a blue or orange circle. The attacker is represented in orange and plays with a period of $\delta_A = 4$. The defender is represented in blue and plays with a period of $\delta_D = 3$. The blue and orange rectangles represent the amount of time the respective player is in control of the resource.

The state of the resource is denoted as a time-dependent variable $C = C_i(t)$. $C_D(t)$ is 1 if the game is under control by the defender and 0 if the game is under control by the attacker. Reversely, $C_A(t)$ will be 1 if the game is under control by the attacker and 0 if under control by the defender. So, $C_A(t) = 1 - C_D(t)$. The game starts with the defender being in control: $C_D(0) = 1$.

The players receive a benefit equal to the time units they were in possession of the resource minus the cost of making their moves. The cost of a player *i* is denoted by k_i. The total gain of player *i* is equal to the total amount of time that a player *i* has owned the resource from the beginning of the game up to time *t*. It is expressed as follows:

$$G_i(t) = \int_0^t C_i(x)dx. \tag{1}$$

If we add up the gain of the defender and the gain of the attacker it should sum up to t:

$$G_D(t) + G_A(t) = t \tag{2}$$

The average gain rate of player *i* is defined as:

$$\gamma_i(t) = G_i(t)/t. \tag{3}$$

And thus for all $t > 0$:

$$\gamma_D(t) + \gamma_A(t) = 1 \tag{4}$$

Let $\beta_i(t)$ denote player's *i* average benefit upto time *t*:

$$\beta_i(t) = \gamma_i(t) - k_i\alpha_i. \tag{5}$$

This is equal to the fraction of time the resource has been owned by player *i*, minus the cost of making the moves. α_i defines the average move rate by player *i* up to time *t*. In a given game, the asymptotic benefit rate (or simply benefit) will be defined as the lim inf of the average benefit because time t will increase to infinity and the average benefit may not have limiting values.

$$\beta_i(t) = \lim_{t\to\infty} inf\beta_i(t) \tag{6}$$

1) strategies: Because the players move in a stealthy way, there are different types of feedback that a player can get while moving. These types of feedback can be divided into two groups of strategies. The non-adaptive strategies and the adaptive strategies. These are described in table I.

If there is no feedback for neither of the players, we have a non-adaptive strategy. Because a player does not receive any feedback during the game he will play in the same manner against every opponent. The strategy is called non-adaptive because the playing strategy is not dependent on the opponents movements. An interesting subclass of the non-adaptive strategies is the one where the time intervals between two consecutive moves are generated by a renewal process. An example of such renewal strategy is the periodic strategy where the time between two consecutive moves of the players are a fixed interval. An exponential strategy is a renewal strategy in which the interval between two consecutive moves is exponentially distributed.
In case there is feedback, a player can adapt his strategy to the information received about the opponent's moves. Depending on the amount of information received, two subclasses of adaptive strategies can be identified. The Last Move (LM) strategies represent the class where whenever a player flips he will find out the exact time that the opponent played

Categories	Classes of Strategies
Non-adaptive (NA)	Renewal - Periodic - Exponential General non-adaptive
Adaptive (AD)	Last move (LM) Full History (FH)

TABLE I: Hierarchy of Classes of strategies in FlipIt

the last time. In the second class, called Full History (FH), whenever a player flips he will find out the whole history of the opponent's move.
In this paper we will focus on the non-adaptive strategies. This choice is motivated by the fact that in a security game a player (defender or attacker) rarely has information about the moves (last move or full history) of his opponent.

The study of the different strategies by means of FlipIt framework allows to derive a number of interesting results:

- periodic games dominate the other renewal strategies, meaning that it is always advantageous to play periodically against an opponent with a renewal strategy;
- periodic games are disadvantageous against players following a Last Move adaptive strategy;
- if the defender plays with a periodic rate that is fast enough he'll force the attacker to drop out;
- any amount of feedback about the opponent received during the game, benefits to a player.

III. FlipIt with virus propagation

A FlipIt game consists of a single resource. To represent the security problem, the game now defines its single resource as a computer network with multiple nodes. One of the players, the defender, will try to defend his network. The defender will do this by flipping all the nodes of the network (i.e. the entire resource) in every move he plays. The attacker, the other player, will try to infect all the nodes in the network. The attacker will do this by flipping the node in the graph that can infect all the nodes in the shortest possible time. After dropping a virus on the first node, it takes a while for the virus to infect the entire network. Since the original FlipIt game works with a single resource that is always flipped entirely, the assumption is made that the attacker is considered to gain immediate full control over the resource when the network has been infected, even it is only one node that has been infected.

In reality however, after dropping a virus on the first node, it takes a while for the virus to infect the entire network. So, the assumption that the attacker has full control over the resource as soon as a node has been infected, is not realistic. The attacker has only control of the network once all or a sufficient number of nodes are infected. The time that it takes for the virus to infect every node (or a sufficient number of nodes) will be denoted as an infection-delay variable *d* (called 'delay' for short in the remainder of this paper). If we want to measure how long it takes for the virus to infect all the nodes in the network, we have to calculate the shortest path from the first infected node to the farthest node. Rather than denoting the time needed for infecting *all* the nodes, the variable d can also be used to denote the time needed to infect *a sufficient number* of nodes.

Assume that an attacker attacks at time *t*, he doesn't get immediate control over the resource, but he only gains control at time *t* + *d*, with d denoting the time needed to infect a sufficiently number (or all) nodes. If the defender flips the network before the period d has elapsed (so, somewhere between t and $t + d$), then the attacker will never gain full control over the resource. This implies that the mathematical formulas for gain and benefit need to be adapted to the fact that the attacker loses part of its benefit because of this delay. In the remainder of this paper, we will adapt the formalization of the FlipIt game using the variable d.

The formalization starts from the model of the non-adaptive continuous basic FlipIt game where players use a periodic strategy with a random phase. This choice is motivated by the assumption that in most organisations, the defence strategy is to periodically defend the network. This corresponds to a periodic defender strategy. A periodic attacker strategy is assumed as well, to be able to compare the results with the periodic strategy of the FlipIt game in [2]. Further research can investigate the effect of relaxing this assumption.

Similarly as in [2], we split the formalization in two cases. The first case is where the defender plays at least as fast as the attacker, the second case is where the attacker plays at least as fast as the defender. For each of these cases, first the benefit formula of the basic case without delay is presented, and then the delay is introduced.

A. Formalization the benefit formula including the infection-delay

A Periodic strategy is a non-adaptive renewal strategy where the time intervals between consecutive moves are a fixed period, denoted by δ. Moreover it has a random phase, that is chosen uniformly and random in the interval $[0, \delta]$ for the first move. The average rate of play of a player is denoted by $\alpha_i = \frac{1}{\delta_i}$.

Case 1: $\delta_D \leq \delta_A$ *(The defender plays at least as fast as the attacker.)*

Let $r = \frac{\delta_D}{\delta_A}$. The intervals between two consecutive defender's moves have length δ_D. Consider a given defender move interval. The probability over the attacker's phase selection that the attacker moves in this interval is r. Given that the attacker moves within the interval, he moves exactly once within the interval (since $\delta_D \leq \delta_A$) and his move is distributed uniformly at random.

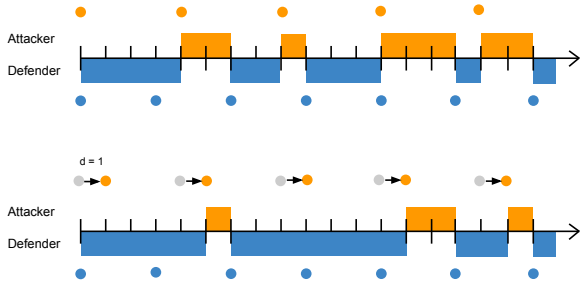
The expected period of attacker control within the interval would be r/2, without considering the delay by a virus. Therefore the benefit for the attacker, without considering the delay, can be expressed as follows:

$$\beta_A(\alpha_D, \alpha_A) = \frac{r}{2} - k_A\alpha_A = \frac{\delta_D}{2\delta_A} - k_A\alpha_A \quad (7)$$

Correspondingly, the benefit for the defender can be expressed as:

$$\beta_D(\alpha_D, \alpha_A) = 1 - \frac{r}{2} - k_D\alpha_D = 1 - \frac{\delta_D}{2\delta_A} - k_D\alpha_D \quad (8)$$

Fig. 2: The first FlipIt game is one without virus propagation. The second one is with virus propagation and d = 1. The delay is denoted with an arrow.



However, because of the delay required for virus propagation, the maximal time of control is reduced to $\delta_D - d$, see figure 2. There is a probability of r that the attacker will move in the interval of the defender. However, the gain will not be half of the interval. Indeed, the attacker has to play soon enough to gain control, meaning that the attacker has to play during the period of $\delta_D - d$ during the interval of the defender. The probability that the attacker plays soon enough is $\frac{\delta_D - d}{\delta_D}$ and this will give the attacker an average gain of $\frac{\delta_D - d}{2}$. If the attacker moves after the period of $\delta_D - d$, the gain of the attacker will be zero. The probability that this happens is $\frac{d}{\delta_D}$. The average gain rate of the attacker can then be expressed as follows if we look at one interval of the defender:

$$\gamma_A(\alpha_D, \alpha_A) = \frac{1}{\delta_D}[\frac{\delta_D}{\delta_A} \cdot \frac{\delta_D - d}{\delta_D} \cdot \frac{\delta_D - d}{2} + \frac{\delta_D}{\delta_A} \cdot \frac{d}{\delta_D} \cdot 0] \quad (9)$$

To derive the benefit, the cost of moving is subtracted from the average gain.

$$\beta_A(\alpha_D, \alpha_A) = \frac{(\delta_D - d)^2}{2 \cdot \delta_D\delta_A} - k_A\alpha_A \quad (10)$$

$$\beta_A(\alpha_D, \alpha_A) = \frac{\delta_D}{2 \cdot \delta_A} - k_A\alpha_A - (\frac{d^2}{2 \cdot \delta_A\delta_D} - \frac{d}{\delta_A}) \quad (11)$$

The benefit of the defender is expressed as follows:

$$\beta_D(\alpha_D, \alpha_A) = 1 - \frac{(\delta_D - d)^2}{2 \cdot \delta_D\delta_A} - k_D\alpha_D \quad (12)$$

We can easily see that when d=0, we obtain the formula of the original FlipIt game.

Case 2: $\delta_A \leq \delta_D$ *(The attacker plays at least as fast as the defender.)*

First let $r = \frac{\delta_D}{\delta_A}$. The intervals between two consecutive attacker's moves have length δ_A. Consider a given attackers move interval. The probability over the attacker's phase selection that the defender moves in this interval is $\frac{\delta_A}{\delta_D} = (1/r)$. Given that the defender moves within the interval of the attacker, he moves exactly once within this interval (since $\delta_A \leq \delta_D$) and his move is distributed uniformly at random.

A similar analysis as in case 1 for a FlipIt game without virus propagation yields the following benefits:

$$\beta_D(\alpha_D, \alpha_A) = \frac{1}{2r} - k_D\alpha_D = \frac{\delta_A}{2\delta_D} - k_D\alpha_D \quad (13)$$

$$\beta_A(\alpha_D, \alpha_A) = 1 - \frac{1}{2r} - k_A\alpha_A = 1 - \frac{\delta_A}{2\delta_D} - k_A\alpha_A \quad (14)$$

For the case with a virus we consider two cases, Case a and Case b, depending on whether the delay is shorter or longer than the difference between the attacker's and the defender's period.

Case a: $d + \delta_A \leq \delta_D$*:*
Consider a timespan $\delta_A + d$, representing the attacker's interval followed by the delay period in his next interval. The defender will never move twice during this timespan because $\delta_A + d \leq \delta_D$. The defender will move during the interval of the attacker with a probability of $\frac{\delta_A}{\delta_D}$. When this happens the defender will end with being in control at the end of the interval. In the next interval the attacker will have to regain control, meaning that during the delay, the defender stays in control, see figure 3 cases (1) and (2). This means that the defender will keep the control over the resource in the next interval over a period of the delay, namely d. Because $d + \delta_A \leq \delta_D$ the next move of the defender in this second interval will never occur during the delay, meaning that the entire delay can be considered as an extra benefit resulting of a play in the previous interval. So, every time the defender plays, he will get an average gain of $\frac{\delta_A}{2}$ in the interval where he plays and in the next interval will always receive a extra gain of d, yielding a total average gain per interval of $\frac{(d + \frac{\delta_A}{2})}{\delta_A}$

The total gain rate of the defender is then the probability that the defender will move during an interval of the attacker multiplied by the total average gain per interval:

$$\gamma_D(\alpha_D, \alpha_A) = \frac{\delta_A}{\delta_D} \cdot \frac{(d + \frac{\delta_A}{2})}{\delta_A} \tag{15}$$

$$\gamma_D(\alpha_D, \alpha_A) = \frac{\delta_A}{2\delta_D} + \frac{d}{\delta_D} \tag{16}$$

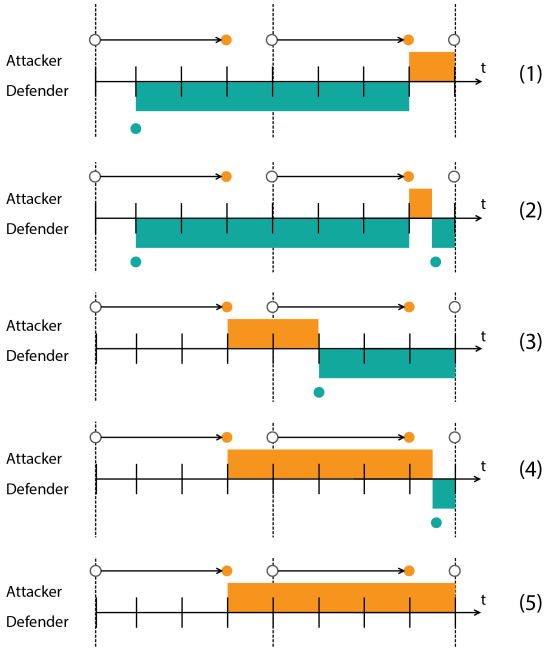
This yields in the following benefit formula:

$$\beta_D(\alpha_D, \alpha_A) = \frac{\delta_A}{2\delta_D} + \frac{d}{\delta_D} - k_D\alpha_D \tag{17}$$

The benefit for the attacker will be as follows:

$$\beta_A(\alpha_D, \alpha_A) = 1 - \frac{\delta_A}{2\delta_D} - \frac{d}{\delta_D} - k_A\alpha_A \tag{18}$$

Fig. 3: Case 2 where d + deltaA < deltaD



It is crucial that δ_D is at least as large as $d+\delta_A$. If not, this would mean that the defender can move during the delay in the interval following the interval where the defender already moved. This would mean that there can be an overlap between the average gain of $\frac{\delta_A}{2}$ and the delay. The above benefit formula would then include to much gain for the defender: the potential overlap during the delay would be counted twice.

Case b: $d + \delta_A \geq \delta_D$*:*

To obtain the formula in case of a too long delay, we therefore need to subtract this overlapping gain from the above formula. Since $\delta_D \geq \delta_A$, if the defender enters the interval immediately after the attacker has played, then the defender cannot have played in the previous interval. In that case, there is no overlap. So the problem of the overlap only appears if the defenders enters late enough and thus only the last part of the delay is subject to overlap. The larger the difference between the interval of the defender and the attacker, the smaller the risk of overlap. Concretely, only the last part of length $d-(\delta_D-\delta_A)$ is subject to overlap. Hence, the probability of overlap is $\frac{d-(\delta_D-\delta_A)}{\delta_D}$ and the gain will be half of this interval: $\frac{d-(\delta_D-\delta_A)}{2}$. The gain rate to be subtracted is therefore:

$$\frac{1}{\delta_A} \cdot \frac{d-(\delta_D-\delta_A)}{\delta_D} \cdot \frac{d-(\delta_D-\delta_A)}{\delta_D} \tag{19}$$

The total gain rate of the defender is obtained by subtracting this term from the gain rate of case a:

$$\gamma_D(\alpha_D, \alpha_A) = \frac{\delta_A}{\delta_D} \cdot \frac{(d + \frac{\delta_A}{2})}{\delta_A} - \frac{(d-(\delta_D-\delta_A))^2}{2\delta_D\delta_A} \tag{20}$$

$$\gamma_D(\alpha_D, \alpha_A) = \frac{\delta_A}{2\delta_D} + \frac{d}{\delta_D} - \frac{(d-(\delta_D-\delta_A))^2}{2\delta_D\delta_A} \tag{21}$$

This yields in the following benefit formula:

$$\beta_D(\alpha_D, \alpha_A) = \frac{\delta_A}{2\delta_D} + \frac{d}{\delta_D} - k_D\alpha_D - \frac{(d-(\delta_D-\delta_A))^2}{2\delta_D\delta_A} \tag{22}$$

The benefit for the attacker will be as follows:

$$\beta_A(\alpha_D, \alpha_A) = 1 - \frac{\delta_A}{2\delta_D} - \frac{d}{\delta_D} - k_A\alpha_A + \frac{(d-(\delta_D-\delta_A))^2}{2\delta_D\delta_A} \tag{23}$$

IV. Related work on FlipIt

Various possible ways to extend FlipIt have already been proposed. Laszka et al. made a lot of additions and extensions to the original game of FlipIt. For instance Laszka et al. extended the basic FlipIt game to multiple resources. The rationale is that for compromising a system in real life, more than just one resource needs to be taken over. An example is that gaining access to deeper layers of a system may require breaking several passwords. The model is called FlipThem [3]. Laszka et al. also use two ways to flip the multiple resources: the AND and the OR control model. In the AND model the attacker only controls the system if he controls all the resources of the

system, whereas in the OR model the attacker only needs to compromise one resource to be in control of the entire system.

Another addition of Laszka et al. to the game of FlipIt [4] is extending the game to also consider non-targeted attacks by non-strategic players. In this game the defender tries to maintain control over the resource that is subjected to both targeted and non-targeted attacks. Non-targeted attacks can include phishing, while targeted attacks may include threats delivered through zero day attack vulnerabilities.
One of the last important additions from Laszka et al. [5] is to consider a game where the moves made by the attacker are still covert but the moves made by the defender are known to the attacker. This means that the attacker can base his attacks on the defender's moves. Both the targeted and non-targeted attacks don't succeed immediately. For the targeted attack the time till it succeeds is given by an exponential distributed random variable with a known rate. The non-targeted attacks are modelled as a single attacker and the time till it succeeds is given by a Poisson process. The conclusion of this paper is that the optimal strategy for the defender is moving periodically. The difference with this paper is that the delay in this paper is dependent on the number of nodes that have to be flipped in a network. This cannot be modelled with the framework in the Laszka et al. paper because the delay is chosen as an exponential distributed random variable. Another difference is that in the case of the Laszka et al. paper, the moves of the defender are considered not stealthy and so the attacker knows when the defender plays.

Other authors used the FlipIt game to apply it on a specific scenario. To be able to use the FlipIt game, modifications where required for the FlipIt model. One of the scenarios by Pham [6] was to find out whether a resource was compromised or not by the attacker. This could be verified by the defender, who has an extra move "test" beside the flip move. The basic idea is to test with an extra action if the resource has been compromised or not. This move involves also an extra cost.

Finally researchers also have investigated the behaviour of humans playing FlipIt. A. Nochenson and Grossklags [7] investigate how people really act when given temporal decisions. Reitter et al. [8] extended the work of A. Nochenson and Grossklags to include various visual presentation modalities for the available feedback during the investigation.

V. Conclusions and Further Research

In this paper we presented an adaptation of the FlipIt game to the situation of virus propagation, such as to take the delay for network infection into account. We discerned two cases. In the case the defender plays faster than the attacker, the attacker simply looses the delay. In the case the attacker plays faster, each time the defender plays in an interval, he will gain extra time of the delay. The delay is therefore always detrimental to the benefit of the attacker. This demonstrates that the FlipIt game can be adapted to a game with virus propagation. Further research needs to be performed to calculate the impact of the delay on Nash equilibria and the determination of optimal defender and attacker strategies.

References

[1] "Advanced persistant threats (apt)." [Online]. Available: https://www.schneier.com/blog/archives/2011/11/advanced_persis.html

[2] M. van Dijk, A. Juels, A. Oprea, and R. Rivest, "Flipit: The game of "stealthy takeover"," *Journal of Cryptology*, vol. 26, no. 4, pp. 655–713, 2013. [Online]. Available: http://dx.doi.org/10.1007/s00145-012-9134-5

[3] A. Laszka, "Flipthem: Modeling targeted attacks with flipit for multiple resources," *5th International Conference, GameSec 2014, Los Angeles, CA, USA, November 6-7, 2014. Proceedings*, vol. 8840, pp. 175–194, 2014.

[4] A. Laszka, B. Johnson, and J. Grossklags, "Mitigating covert compromises," *iets*, vol. 8289, pp. 319–332, 2013. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-45046-4-26

[5] ——, "Mitigation of targeted and non-targeted covert attacks as a timing game," vol. 8252, pp. 175–191, 2013. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-02786-9-11

[6] V. Pham and C. Cid, "Are we compromised? modelling security assessment games," in *Decision and Game Theory for Security*, ser. Lecture Notes in Computer Science, J. Grossklags and J. Walrand, Eds. Springer Berlin Heidelberg, 2012, vol. 7638, pp. 234–247. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34266-0_14

[7] A. Nochenson, J. Grossklags *et al.*, "A behavioral investigation of the flipit game," in *Proceedings of the 12th Workshop on the Economics of Information Security (WEIS)*, 2013.

[8] D. Reitter, J. Grossklags, and A. Nochenson, "Risk-seeking in a continuous game of timing," in *Proceedings of the 13th International Conference on Cognitive Modeling (ICCM)*, 2013, pp. 397–403.

Appendix B

The Last Appendix

SPELTHEORIE EN CYBERSECURITY

Een studie over strategieën voor het verdedigen van bedrijfsnetwerken

Sophie Marien

Virussen zijn een groot probleem voor bedrijfsnetwerken. Ze kunnen gevoelige informatie verzamelen of een bedrijfsnetwerk platleggen. Gegeven de grote kost gebonden aan schade door malware, is het vinden van de juiste verdedigingsstrategie belangrijk. Het aanvallen en verdedigen van een bedrijfsnetwerk kan gezien worden als een spel, waarbij de verdediger en de aanvaller elk proberen de beste strategie te vinden. In dit artikel lichten we toe hoe het spel van aanvallen en verdedigen kan gedefinieerd worden als een variatie op het spel FlipIt. Dit laat toe om te onderzoeken wat de verschillende strategieën zijn van de netwerkbeheerder enerzijds en van de aanvaller die virussen zendt anderzijds. De bedoeling is om in een volgende stap het spel verder te analyseren met behulp van speltheorie om te bepalen welke de dominerende strategieën zijn en of er zich Nash equilibria voordoen.

Security is het geheel van middelen die ingezet worden om een doel te beveiligen tegen kwaadaardige bedreigingen. Deze bedreigingen variëren van virussen die programma's installeren, tot het lekken van vertrouwelijke informatie of een programma voor een '*denial of service' attack*. De jaarlijkse kost voor een bedrijf aan security kan hoog oplopen en daarom is het dus belangrijk voor een bedrijf om de juiste verdedigingsstrategie te vinden.

CYBERSECURITY

In dit artikel concentreren we ons op cybersecurity. Cybersecurity is een onderdeel van security en focust zich op het beveiligen van computergestuurde apparaten zoals computers en smartphones, evenals computernetwerken zoals publieke en private netwerken, met inbegrip van het hele internet. Een privaat netwerk zoals een bedrijfsnetwerk wordt afgeschermd van het publiek netwerk zoals het internet. Het doel van beveiliging is zekerheid te geven dat data niet wordt verwijderd zonder toelating (confidentialiteit), dat de data altijd toegankelijk is (beschikbaarheid) en dat de data niet wordt gelezen of gewijzigd door iemand die hier geen toelating voor heeft (integriteit).

Om te weten hoe een systeem verdedigd moet worden, is het belangrijk om te weten hoe het aangevallen kan worden. Een van de manieren om een systeem of computer aan te vallen is door gebruik te maken van malware. Dit is een kwaadwillig stuk programma dat zal proberen

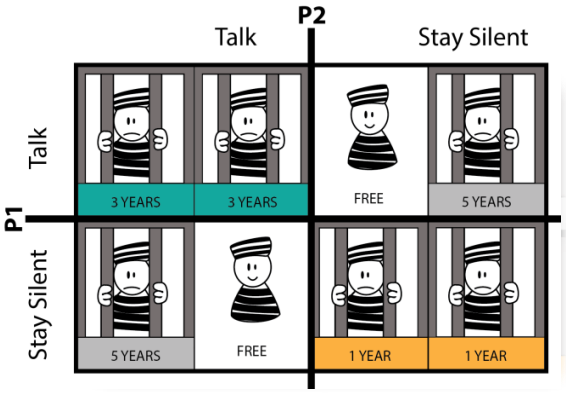
om onbeveiligde systemen of computers binnen te dringen en daar aan gevoelige informatie te geraken. Virussen, wormen, trojans zijn voorbeelden van malware.

SPELTHEORIE EN CYBERSECURITY

Speltheorie kan op verschillende domeinen toegepast worden. Denk maar aan politiek, economie, biologie, sociologie … en ook op het domein van security. Speltheorie bestudeert de strategische interactie tussen de spelers in een spel. In een spel kunnen er een aantal spelers zijn die elk acties kunnen uitspelen. Deze acties zijn voorgesteld door een getal dat hun voorkeur aangeeft.

GEVANGENEN DILEMMA

Een voorbeeld van een spel met twee spelers is bijvoorbeeld het gevangenisdilemma. In dit spel zijn er twee spelers die beiden rationeel zijn en



Figuur 1: Het gevangenisdilemma: P1 komt overeen met de eerste kolom, P2 met de tweede kolom.

beiden een misdaad hebben begaan. 'Rationeel zijn' betekent dat ze het beste voor zichzelf willen en het niet hun doel is om de ander kwaad aan te doen. Allebei zitten ze opgesloten in een apart lokaal en weten ze niet van elkaar wat ze gaan vertellen. Elk van hen kan de ander verraden of ze kunnen elkaar steunen en blijven zwijgen. Als een speler bekent, krijgt hij afhankelijk van wat de andere doet, drie jaar gevangenis of hij is vrij om te gaan. Als de speler zwijgt krijg hij afhankelijk van wat de andere speler doet ofwel vijf jaar gevangenis ofwel een jaar.

SPELTHEORIE KAN OP VERSCHILLENDE DOMEINEN TOEGEPAST WORDEN .. OOK OP HET DOMEIN VAN SECURITY

Figuur 1 toont de verschillende combinaties van de mogelijke acties van de twee spelers. Het Nash equilibrium (zie verder) van het spel is dat ze allebei zwijgen, maar perfect rationele spelers kiezen er toch voor om allebei te bekennen. Dit komt omdat dit de dominante strategie is. Een dominante strategie is een strategie die beter is als alle andere strategieën van een speler onafhankelijk van de tegenspeler. Hier is het voor elke speler voordeliger om te bekennen. Dit levert hen drie jaar gevangenis op in plaats van een jaar. Waarom spelers voor de ene of de andere actie kiezen kan uitgelegd worden aan de hand van speltheorie.

Door speltheorie te gebruiken kan men uitzoeken hoe een netwerk het best kan verdedigd worden tegen aanvallers. Het spel dat gemodelleerd wordt, is een spel tussen twee spelers: de verdediger en de aanvaller. De verdediger kan de netwerkmanager zijn die het netwerk van een bedrijf zal moeten verdedigen. De aanvaller kan een programmeur zijn die virussen schrijft om een netwerk van het bedrijf aan te vallen.

FLIPIT

In dit artikel bespreken we een bepaald model om dit soort spelen te modelleren, met name FlipIt (figuur 2). FlipIt is een spel dat gespeeld wordt door twee spelers, de verdediger en de aanvaller. Beiden willen de controle krijgen over een gemeenschappelijke *resource*. Deze *resource* kan bijvoorbeeld een wachtwoord, een computer of een volledig netwerk zijn.

De spelers kunnen de controle krijgen door de *resource* te flippen. Met flippen wordt er een actie uitgevoerd. Dus als de verdediger de *resource* flipt dan heeft hij de controle over de *resource*. Als de aanvaller erna de *resource* flipt dan verliest de verdediger de controle over de *resource* en heeft de aanvaller nu de controle over de *resource*. Een flip kan op elk moment gebeuren. De spelers moeten niet tegelijkertijd spelen of eerst wachten op een actie van de andere speler. Er moet ook rekening mee gehouden worden dat elke flip een bepaalde kost inhoud. FlipIt is een spel dat oneindig lang doorgaat. Het doel van het spel is voor elke speler om de tijd dat hij de *resource* in bezit heeft te maximaliseren en zijn kost te

Figuur 2: FlipIt

minimaliseren. Wat FlipIt anders maakt dan de andere spelen in speltheorie is dat het flippen *stealthy* gebeurt. Er wordt dus heimelijk geflipt, wat betekent dat de andere speler niet weet wanneer zijn tegenspeler de controle over de *resource* probeert over te nemen. Het kan voorvallen dat een speler denkt dat hij de controle over de *resource* kwijt is en een flip doet terwijl hij toch nog de controle over de *resource* heeft. Dit wordt dan een “flop” genoemd omdat dit een verloren kost inhoud.

Een kleine toepassing van FlipIt is het beschermen van een *resource* via een wachtwoord. Wanneer de aanvaller het wachtwoord reset, wat overeenkomt met een flip, heeft hij bezit over de *resource.* De verdediger kan dit terug flippen door weer het wachtwoord te resetten. Geen van beide spellers weet wanneer de andere het wachtwoord gereset heeft.

VAN FLIPIT NAAR CYBERSECURITY

Veel bedrijfsnetwerken moeten zich continu verdedigen tegen indringers van buitenaf zoals virussen en wormen. De netwerkbeheerder zal proberen het netwerk zo malware-vrij mogelijk te houden. Als er dan toch een indringer is geslaagd om het netwerk binnen te dringen dan zal de netwerk manager deze indringer zo snel mogelijk proberen buiten te krijgen. Dit is niet altijd even makkelijk. Zeker niet wanneer de indringers heimelijk binnenglippen en zich dan snel verspreiden.

Cybersecurity vertoont dus gelijkenissen met het spel FlipIt. Het flippen komt overeen met het overnemen van de controle over (een deel van) het netwerk. Het gebeurt ook heimelijk en continu. Toch zijn er ook verschillen, te wijten aan de complexiteit van de verschillende vormen van malware. Virussen hebben verschillende manieren om zich te verspreiden en verschillen ook in de schade die ze willen toebrengen.

Het “I love you” virus is een voorbeeld van een virus dat zich snel verspreid. Dit virus plant zich voort via mailsystemen. Als iemand een mail opent met het “I love you” virus in bijlage dan verspreidt dit virus zichzelf door een mail te sturen met zichzelf naar iedereen in de

contactlijst. Zo kan het virus zich zeer snel vermenigvuldigen en uiteindelijk het netwerk van een bedrijf platleggen door het vele verkeer. In dit voorbeeld is er een menselijke interactie nodig om het virus te doen verspreiden. Als niemand de mail opent dan kan het virus zich niet verspreiden.

Jammer genoeg bestaan er ook virussen die zich kunnen verspreiden zonder menselijke interactie. Deze virussen worden wormen genoemd. Een worm is ook een computerprogramma dat zich dupliceert om zich zo te verspreiden naar andere computers. Via een computernetwerk worden kopieën van de worm doorgestuurd zonder dat er een tussenpersoon voor gebruikt wordt. De worm zal gebruikmaken van beveiligingslekken om andere computers te infecteren.

De meeste wormen worden gemaakt om zich

WORMEN ZIJN VIRUSSEN DIE ZICH KUNNEN VERSPREIDEN ZONDER MENSELIJKE INTERACTIE

alleen maar te verspreiden en proberen geen veranderingen aan te brengen aan de systemen die ze passeren. Deze wormen kunnen nog steeds schade toebrengen door de verhoogde netwerktrafiek die ze genereren. Wormen die wel schade berokken bevatten een programma om een *backdoor* te installeren of een *rootkit* op de geïnfecteerde computers. De *backdoors* en *rootkits* zorgen ervoor dat er later gebruik kan gemaakt worden van de geïnfecteerde computers.

De Stuxnetworm is een zeer bekende worm. Initieel verspreide deze worm zich via geïnfecteerde USB sticks en vanaf dan kon het zich via het internet verspreiden naar andere computers. Het doel van de Stuxnetworm was om de centrifuges in kernreactoren kapot te laten draaien. Vele kernreactoren zijn geïnfecteerd geweest. Vanuit het standpunt van de verdediger is het dus zeer belangrijk om zo snel mogelijk te reageren zodat de worm zich niet snel kan verspreiden.

AANPASSINGEN AAN FLIPIT

Om via FlipIt een situatie van aanvallen van virussen en wormen te modelleren zijn er dus een aantal aanpassingen aan FlipIt nodig.

De eerste aanpassing is dat de enkele *resource* wordt vervangen door meerdere *resource*s. Deze stellen de knooppunten voor in het bedrijfsnetwerk. Elk knooppunt is een computer van een werknemer in het bedrijf. De verbindingen (linken) tussen de knooppunten zijn de logische communicatieverbindingen, zoals de contactpersonen in een mailinglijst van de computer. Er wordt van uitgegaan dat als de ene computer iemand in zijn contactlijst heeft staan dat de andere deze ook in zijn contactlijst heeft staan zodat de linken bidirectioneel zijn.

De tweede en laatste aanpassing is een extra actie voor de spelers. In plaats van te flippen is het nu ook mogelijk om te “onderzoeken”. Dat betekent dat de *resource* nog niet geflipt wordt, maar er gekeken wordt wie de controle heeft over de *resource*. De kost voor het “onderzoeken” is minder groot dan de kost voor het flippen. Dit zou kunnen betekenen dat het misschien voordeliger is om eerst na te gaan of een knooppunt geïnfecteerd is en pas daarna flippen als het knooppunt effectief geïnfecteerd is. Wat onveranderd blijft is dat het flippen en het “onderzoeken” steeds heimelijk gebeurt en dat het spel in een continue tijd doorgaat.

Op een gegeven moment zal de aanvaller een virus sturen of plaatsen op een van de knooppunten via bijvoorbeeld een USB stick. De verdediger zal ten alle tijden proberen zijn netwerk *clean* te houden. Vanaf dat moment zal het virus zich gaan verspreiden over de andere knooppunten. De propagatiestrategie van het virus is vooraf bepaald. Dat betekent dat de propagatie snelheid vast ligt en de actie van het virus.

Het virus heeft twee acties die het kan uitspelen. De ene actie is dat het onmiddellijk alle knooppunten waarmee het in verbinding staat gaat infecteren. De andere actie is dat het virus telkens maar één knooppunt kan infecteren. Het virus kan voor deze actie kiezen om minder snel opgemerkt te worden. Een geïnfecteerd knooppunt kan maar een keer al zijn naburige knooppunten infecteren. Een variatie op deze twee acties is dat het al dan niet een wederkerige actie kan zijn. Dit betekent dat een geïnfecteerd knooppunt zijn aanvaller terug kan infecteren. Hierdoor kan dit knooppunt terug al zijn buren infecteren.

De verdediger heeft één belangrijke actie: het flippen of "onderzoeken" van een bepaald aantal knooppunten per keer. De kost van het aantal knooppunten stijgt op een progressieve

SPELTHEORIE IS TOEPASBAAR BINNEN CYBERSECURITY

manier zodat de verdediger niet als triviale zet alle knooppunten flipt. Voor een verdediger is het de bedoeling om het bedrijfsnetwerk *clean* te houden op een zo goedkoop mogelijke manier omdat hij over een bepaalde tijd binnen een budget blijven. Een variatie op deze actie is dat de verdediger ervoor kan kiezen om de knooppunten in groep of onafhankelijk van elkaar te flippen. De enige speler die vooraf aan de start van het spel informatie heeft, is de verdediger. Deze heeft kennis van de topologie van het netwerk.

VERDER ONDERZOEK

Voor het verdere onderzoek kunnen we via FlipIt analyseren wat de dominante en optimale verdedigingsstrategieën zijn voor de verdediger en aanvallingsstrategieën van de aanvaller. Er kan ook onderzocht worden of het spel een Nash equilibrium heeft. Speltheorie is dus toepasbaar binnen cybersecurity en FlipIt leent zich voor speltheoretische analyse van cybersecurity.

Nash Equilibrium en John Nash

John Nash speelde een grote rol in de geschiedenis van de speltheorie. Hij is een van de wiskundigen geweest die speltheorie geformaliseerd heeft. Het Nash equilibrium werd naar hem vernoemd. Een Nash equilibrium wordt gezien als een evenwicht tussen beide spelers zodat ze allebei de beste tactiek kiezen en niet meer veranderen als de andere van tactiek veranderen. John Nash breide de theorie over het Nash equilibrium in een paper nog uit met gemengde strategieën. In 1994 kreeg John Nash samen met twee andere wiskundigen gespecialiseerd op het vlak van speltheorie de Nobelprijs voor de economie op basis van hun prestaties in de niet-coöperatieve speltheorie. Over John Nash is een prachtige film gemaakt, "A Beautiful Mind".

Bibliography

[1] K. Bowers, M. van Dijk, R. Griffin, A. Juels, A. Oprea, R. Rivest, and N. Triandopoulos. Defending against the unknown enemy: Applying flipit to system security. In J. Grossklags and J. Walrand, editors, *Decision and Game Theory for Security*, volume 7638 of *Lecture Notes in Computer Science*, pages 248–263. Springer Berlin Heidelberg, 2012.

[2] Z. Chen, L. Gao, and K. Kwiat. Modeling the spread of active worms. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1890–1900. IEEE, 2003.

[3] X. Feng, Z. Zheng, P. Hu, D. Cansever, and P. Mohapatra. Stealthy attacks meets insider threats: A three-player game model.

[4] A. Kamra, H. Feng, V. Misra, and A. D. Keromytis. The effect of dns delays on worm propagation in an ipv6 internet. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 4, pages 2405–2414. IEEE, 2005.

[5] Kaspersky. A business approach to managing data security threats. In *IT security risk survey 2014*, 2014.

[6] Kaspersky. Special report on mittigation strategies for advanced threats. In *The power of protection*, 2014.

[7] A. Laszka. Flipthem: Modeling targeted attacks with flipit for multiple resources. *5th International Conference, GameSec 2014, Los Angeles, CA, USA, November 6-7, 2014. Proceedings*, 8840:175–194, 2014.

[8] A. Laszka, B. Johnson, and J. Grossklags. Mitigating covert compromises. *iets*, 8289:319–332, 2013.

[9] A. Laszka, B. Johnson, and J. Grossklags. Mitigation of targeted and non-targeted covert attacks as a timing game. 8252:175–191, 2013.

[10] K. Leyton-Brown and Y. Shoham. *Essentials of Game Theory: A Concise, Multidisciplinary Introduction*. Synthesis lectures on artificial intelligence and machine learning. Morgan & Claypool Publishers, 2008.

[11] Y. S. M. Jackson, K. Brown. Coursera game theory. Stanford University and The university of Britsh Columbia, 2004.

[12] Microsoft. security bulletin release.

[13] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. *IEEE Security & Privacy*, (4):33–39, 2003.

[14] A. Nochenson, J. Grossklags, et al. A behavioral investigation of the flipit game. In *Proceedings of the 12th Workshop on the Economics of Information Security (WEIS)*, 2013.

[15] V. Pham and C. Cid. Are we compromised? modelling security assessment games. In J. Grossklags and J. Walrand, editors, *Decision and Game Theory for Security*, volume 7638 of *Lecture Notes in Computer Science*, pages 234–247. Springer Berlin Heidelberg, 2012.

[16] S. Qing and W. Wen. A survey and trends on internet worms. *Computers & Security*, 24(4):334–346, 2005.

[17] D. Reitter, J. Grossklags, and A. Nochenson. Risk-seeking in a continuous game of timing. In *Proceedings of the 13th International Conference on Cognitive Modeling (ICCM)*, pages 397–403, 2013.

[18] D. Salomon. Examples of malware. In *Elements of Computer Security*, Undergraduate Topics in Computer Science, pages 137–149. Springer London, 2010.

[19] G. Serazzi and S. Zanero. Computer virus propagation models. In *Performance Tools and Applications to Networked Systems*, pages 26–50. Springer, 2004.

[20] W. Stallings. *Network security essentials: applications and standards*. Pearson Education India, 2007.

[21] S. Staniford, V. Paxson, N. Weaver, et al. How to own the internet in your spare time. In *USENIX Security Symposium*, pages 149–167, 2002.

[22] A. technica: Dan Goodin. Equation group, 2015-02-16.

[23] L. N. Trefethen and D. Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

[24] M. van Dijk, A. Juels, A. Oprea, and R. Rivest. Flipit: The game of "stealthy takeover". *Journal of Cryptology*, 26(4):655–713, 2013.

[25] J. Von Neumann and O. Morgenstern. *Theory of games and economic behavior*. Oxford UP, 1944.

[26] Y. Wang, S. Wen, Y. Xiang, and W. Zhou. Modeling the propagation of worms in networks: A survey. *Communications Surveys & Tutorials, IEEE*, 16(2):942–960, 2014.

[27] Wikipedia. Adjacency matrix, 2015.

[28] Y. Xiang, X. Fan, and W. Zhu. Propagation of active worms: a survey. *International Journal of Computer Systems Science & Engineering*, 24(3):157–172, 2009.

[29] G. Yan and S. Eidenbenz. Modeling propagation dynamics of bluetooth worms (extended version). *Mobile Computing, IEEE Transactions on*, 8(3):353–368, 2009.

[30] C. C. Zou, D. Towsley, and W. Gong. On the performance of internet worm scanning strategies. *Performance Evaluation*, 63(7):700–723, 2006.

Fiche masterproef

Student: Sophie Marien

Titel: Flip the virus: Modelling targeted attacks using FlipIt with propagation delay

Nederlandse titel: Flip the virus: Modelling targeted attacks using FlipIt with propagation delay

UDC: 621.3

Korte inhoud:
Recently, high profile targeted attacks such as the attack on Belgacom (a major Belgian Telecom), have demonstrated that even the most secure companies can still be compromised, and moreover that such attacks can go undetected for a while. This kind of attack is called an APT, Advanced Persistent Threat, and is designed to secretly penetrate a computer network, collect sensitive data and stay hidden for many years. Companies have every interest to mitigate the risks of an APT and the consequences that it can cause. Because of stealthiness, fighting against this kind of attack requires methods that go beyond the standard tools against malware.
A group of researchers at the RSA, van Dijk et al., proposed the game FlipIt (The game of "stealthy takeover") to model stealthy takeovers. It is a 2-players game composed of a single attacker, a single defender and a single shared resource. The players will compete to get control over the shared resource. Every move of the players will involve a cost and these moves happen in a stealthy way. The objective of the game for each player is to maximise the fraction of time being in control of the resource and to minimise the total move cost.
FlipIt does however not take into account that a move may not be instantaneous, but may have a certain delay. We adapt FlipIt such that we can use it to model the game of defending a company network that is attacked by an APT. The FlipIt formulas are adapted such as to take the delay for an APT propagation into account, which in our case will be a delay for the attacker. In this paper, we restrict ourselves to games where both the defender and the attacker play with a periodic strategy. The goal of this paper is to find out if modelling such situations with FlipIt with propagation delay allows us to draw interesting lessons about security measures against APTs.

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdspecialisatie Veilige software

Promotor: Prof. dr. T. Holvoet

Assessoren: Prof. dr. B. Jacobs
Dr. ir. A. Dries

Begeleiders: Ir. Jonathan Merlevede,
Ir. Kristof Coninx