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

© Copyright KU Leuven

Without written permission of the thesis supervisor and the author it is forbidden to reproduce or adapt in any form or by any means any part of this publication. Requests for obtaining the right to reproduce or utilize parts of this publication should be addressed to the Departement Computerwetenschappen, Celestijnenlaan 200A bus 2402, B-3001 Heverlee, +32-16-327700 or by email `info@cs.kuleuven.be`.

A written permission of the thesis supervisor is also required to use the methods, products, schematics and programs described in this work for industrial or commercial use, and for submitting this publication in scientific contests.

Zonder voorafgaande schriftelijke toestemming van zowel de promotor als de auteur is overnemen, kopiëren, gebruiken of realiseren van deze uitgave of gedeelten ervan verboden. Voor aanvragen tot of informatie i.v.m. het overnemen en/of gebruik en/of realisatie van gedeelten uit deze publicatie, wend u tot het Departement Computerwetenschappen, Celestijnenlaan 200A bus 2402, B-3001 Heverlee, +32-16-327700 of via e-mail `info@cs.kuleuven.be`.

Voorafgaande schriftelijke toestemming van de promotor is eveneens vereist voor het aanwenden van de in deze masterproef beschreven (originele) methoden, producten, schakelingen en programma's voor industrieel of commercieel nut en voor de inzending van deze publicatie ter deelname aan wetenschappelijke prijzen of wedstrijden.

Preface

Thank you !

Sophie Marien

Contents

Todo list

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 that moreover that such attacks can go undetected for a while. This kind of attack is called APT, Advanced Persistent Threats, and 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 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 a virus. The FlipIt formulas are adapted such as to take the delay for virus 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.

Results toevoegen

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 voor een bepaalde 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.
Een van de bedreigingen die interessant is voor deze thesis is een Advanced Persistent Threat (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 computer 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 is, maar dat dit 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 dat het toepasbaar is voor heimelijke aanvallen die onderhevig zijn aan een vertraging.

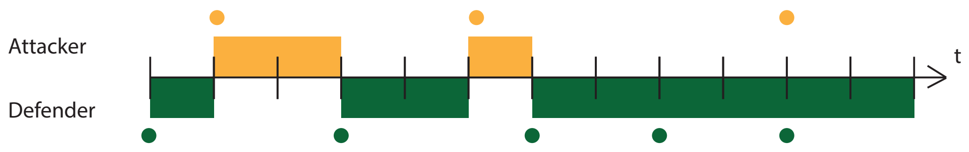
FlipIt

Om te begrijpen hoe we het FlipIt spel kunnen aanpassen om virus propagatie in acht te nemen, is het belangrijk om vertrouwd te raken met de concepten van het basische 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 gemiddeld aantal flippen definieert door speler i met $n_i(t)$ het aantal bewegingen door speler i:

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

Dit 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.

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

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 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.

tabel

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;

- periodieke spellen zijn 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 van 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 dat de speler 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 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].

Net zoals in FlipIt [24], delen we de formalisering in twee gevallen. In het eerste geval speelt de verdediger minstens zo snel als de aanvaller en in het tweede geval speelt de aanvaller minstens zo snel als de verdediger. Voor beide gevallen presenteren we eerst de *gain* formule van een FlipIt spel zonder vertraging, en daarna introduceren we de vertraging.

Formalisering van de benefit formule 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: $\alpha_i = \dfrac{1}{\delta_i}$.

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

Stel $r = \dfrac{\delta_D}{\delta_A}$. The intervallen tussen twee opeenvolgende bewegingen hebben lengte δ_D. Beschouw een interval van de verdediger. De waarschijnlijkheids-verdeling over de fase van de aanvaller dat de aanvaller beweegt in het interval is r. De aanvaller zal exact een keer bewegen in het interval, gegeven dat hij beweegt in dit interval (omdat $\delta_D \leq \delta_A$) en omdat zijn bewegingen uniform en random gedistribueerd zijn.

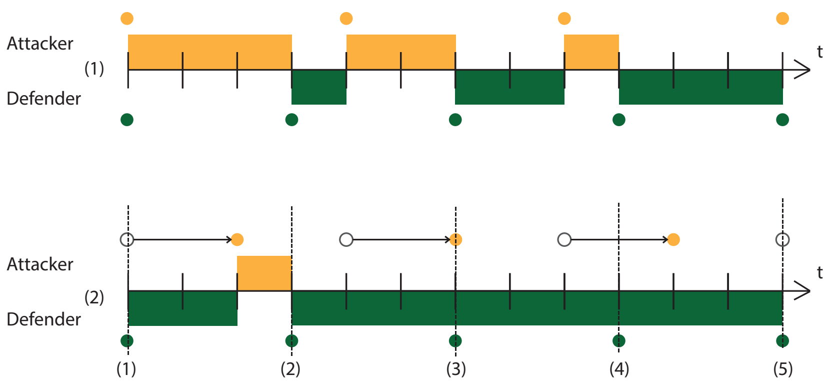
De verwachte periode dat de aanvaller controle heeft in dit interval is *r/2*, zonder de vertraging door een APT in acht te nemen. Daarom kan de benefit voor de aanvaller, zonder de vertraging als volgt uitgeschreven worden:

$$\beta_A(\alpha_D, \alpha_A) = \frac{r}{2} - k_A\alpha_A = \frac{\delta_D}{2\delta_A} - k_A\alpha_A \tag{8}$$

Bijgevolg is de benefit voor de verdediger:

$$\beta_D(\alpha_D, \alpha_A) = 1 - \frac{r}{2} - k_D\alpha_D = 1 - \frac{\delta_D}{2\delta_A} - k_D\alpha_D \tag{9}$$

Echter, omwille van de vertraging die nodig is door de APT propagatie, is de maximale tijd dat de aanvaller in controle kan zijn verminderd naar $\delta_D - d$. Zie figuur 3.5. Er is een kans van r dat de aanvaller zal flippen in het interval van



Figuur 2: 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.

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:

$$\gamma_A(\alpha_D, \alpha_A) = \frac{1}{\delta_D}\left[\frac{\delta_D}{\delta_A} \cdot \frac{\delta_D - d}{\delta_D} \cdot \frac{\delta_D - d}{2} + \frac{\delta_D}{\delta_A} \cdot \frac{d}{\delta_D} \cdot 0\right] \tag{10}$$

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

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

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

De benefit van de verdediger wordt dan als volgt uitgedrukt:

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

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 eerst $r = \frac{\delta_D}{\delta_A}$. De lengte van de intervallen tussen twee opeenvolgende bewegingen van de aanvaller zijn gelijk aan δ_A. Beschouw een gegeven interval van de attacker. 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 APT propagatie 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 APT 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 gedurende 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* van $\frac{\delta_A}{2}$ in het interval waar hij flipt en een extra *gain* van d van 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 verdegier zal flippen gedurende het interval van de aanvaller vermenigvuldigd met het 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:

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

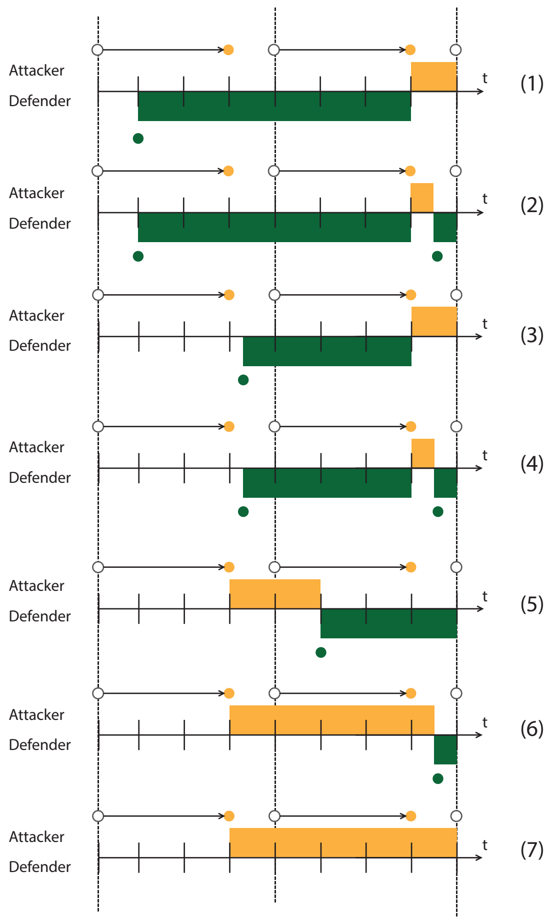
De benefit van de aanvaller is dan als volgt:

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

Het is cruciaal dat δ_D op zen minst zo groot is als $d + \delta_A$. Zoniet betekent dit dat de aanvaller kan bewegingen tijdens de vertraging in het interval die 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.

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

Om de formule te bekomen voor de case waarbij de delay te groot is, moeten we de overlap met de *gain* berekenen en deze aftrekken. Aangezien $\delta_D \geq \delta_A$, als de verdediger onmiddelijk speelt nadat de aanvaller geflipt heeft, van 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



Figuur 3: Case 2.a waar $d + \delta_A < \delta_D$

tot overlap. Hoe groter het verschil tussen de periode van de aanvaller en van de defender, 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 $\frac{d - (\delta_D - \delta_A)}{\delta_D}$ en dat de *gain* de helft zal zijn van dit interval: $\frac{d - (\delta_D - \delta_A)}{2}$. De *gain* rate die tot overlap zorgt en dus afgetrokken moet worden is:

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

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:

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

Nash Equilibrium

Na het bepalen van de benefit functies voor de verdediger en de aanvaller, zijn we geinteresseerd in het vinden van de Nash evenwichten. Dit zijn de evenwichten waarbij de spelers benefit niet gaat vergroten door te veranderen van de speel frequentie. Hiervoor moeten eerst de optimale strategiën bepaald worden. Uit deze optimale strategiën kan het Nash evenwicht bepaald worden.

Meer formeel: een Nash evenwicht is een punt (δ_D^*, δ_A^*) zodat de verdediger zijn benefit $\beta_D(\delta_D, \delta_A^*)$ maximaal is op $\delta_A = \delta_A^*$ en de attacker zijn benefit $\beta_A(\delta_D^*, \delta_A)$ maximaal is op $\delta_D = \delta_D^*$. Eerst een aantal belangrijke notaties. $opt_D(\delta_A)$ is the verzameling van waarden (speeltempo δ_D) die de benefit van de defender optimaliseren voor een vaste waarde van δ_A van de aanvaller. Hetzelfde voor $opt_A(\delta_D)$ die verzameling voorstelt van waarden (speeltempo δ_A) die de benefit van de aanvaller optimaliseren voor een vaste waarde van δ_D van de verdediger.

De benefit functies voor elke case worden eerst nog even opgelijst:

- **Case 1:** $\delta_D \leq \delta_A$

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

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

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

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

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

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

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

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

De functie $opt_D(\delta_A)$:

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

De functie $opt_A(\delta_D)$:

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

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.

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.
$n_i(t)$	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 APT's: 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 [5]. 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 them 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 Flip-It 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 optimum functions and Nash equilibria.
- Based on our results we can give practical recommendations to take measures against advanced attacks. The modelling of FlipIt with a propagation delay will give a guideline for both the attacker and the defender on how to respectively plan attacks or network clean-ups.
- This work includes an overview of different techniques used to calculate the propagation delay of malware (depending on the network layout).
- It presents a method to calculate the speed of the propagation of a worm in a network independent of the topology of the network.

punt herschrijven en meer nadruk leggen op het vinden van de Nash equilibria

en het wordt ook toegepast op de formules

While it may seem trivial to extend the basic FlipIt model with a propagation delay, its mathematical treatment is not. In the paper of Laznka et all. [9] it seems that even a small extension adds to the already significant mathematical complexity. Even though the FlipIt game is quite symmetric, the mathematical complexity rises by adding a propagation delay.

Overview of the thesis

The organisation of this paper is the following. An introduction to cyber security and game theory is given in chapter 2 to allow 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 model to model a FlipIt game with virus propagation. Subsequently formulas are derived to model a FlipIt game with a virus propagation for the specific case where players play a periodic strategy with a random phase.
In Chapter 4 the formulas are further analysed to determine if there is a Nash equilibrium. 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. Finally chapter 6 discusses the main results and provides directions for further research.

chapters kloppen nog niet

Chapter 2

General context

This chapter provides the reader with an introduction to the general context of the work presented in this paper. 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 reader to 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 subcategory. 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 sub category 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 2014 [7] 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 [22]. 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 [8], 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 APT. 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 [7] 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 [12] and [13] is given in this section. For a more detailed and complete introduction to game theory, the reader is referred to [12].
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 does not necessarily mean that they want to harm other agents or that they only care about themselves. Instead it means that each agent has preferences about which states of the world he likes. These preferences are mapped to natural numbers and are called the utility functions. The numbers are interpreted as a mathematical measure that tells 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.

Games can be divided into two types of games: cooperative or non-cooperative. In a non cooperative game, 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 game theory*.

utility, pay-off, outcomes

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 maximizes 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 maximizes 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, 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 (betraying the other). 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.

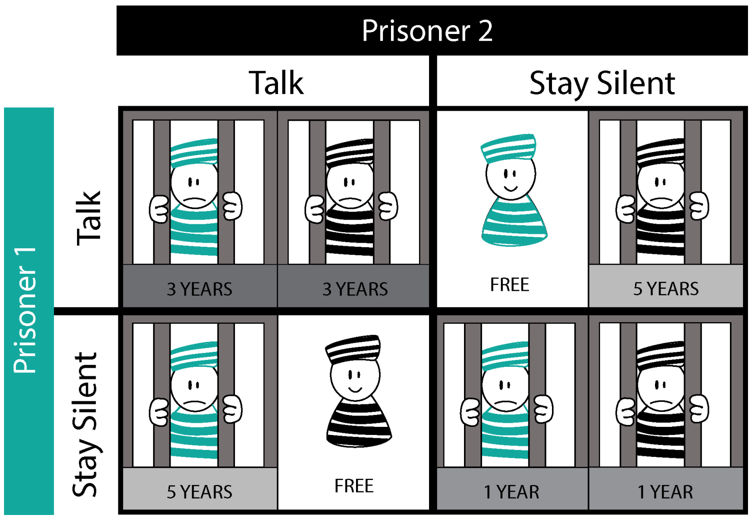


Figure 2.1: Prisoners dilemma: An example from the field of Game Theory. It is a game between two perfectly rational prisoners who do not know what the other one will do. Each prisoner can talk (betraying the other) or remain silent. The resulting sentence is shown below each prisoner.

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-players 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 flip 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 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 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.

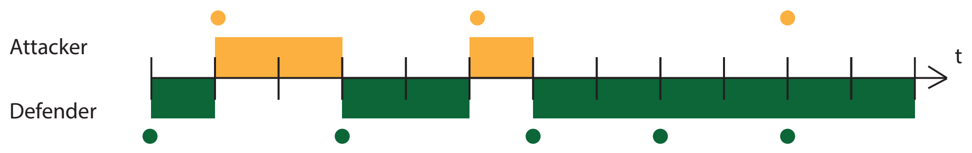


Figure 2.2: A representation of a FlipIt game where both players are playing periodically. Every move or flip is indicated by a green (dark grey) or orange (light grey) circle. The attacker is represented in orange and the defender is represented in green. The blue and orange rectangles represent which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

check comment in latex

If the players receive feedback, a player can adapt his strategy to the information received about the opponent's moves. This strategy is called *adaptive* strategies. Depending on the amount of information received, two subclasses of adaptive strategies can be identified. The Last Move (LM) strategies represent the class where,

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

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 FlipIt framework allows to derive a number of interesting results [24]:

- 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 Last Move 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 [9]. 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 [10] 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. [11] 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 ATP's 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 Laska has multiple attackers who try to find the best strategy of the defender against both targeted and non-targeted attacks. The conclusion of this 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 [2].
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 [17] 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 [5].

Finally, researchers have also investigated the behaviour of humans playing FlipIt. A. Nochenson and Grossklags [16] 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. [19] 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

3.1 Introduction

The FlipIt game with propagation delay considers a game where the moves of the attacker are not instantaneous. This corresponds to APT's 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 3.2 explains the difference between a basic FlipIt game and a FlipIt game with propagation delay. The last section 3.3 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.

3.2 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.

3.2.1 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.

3.2.2 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. In this paper, 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 placing anti-virus systems in a graph. This problem is known to be NP-hard. [1].

niet volledig juist, defender systemen geplaatst en dan gezien hoe ver de worm kan geraken

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 worm to the system. The attacker does not send the same malware again, because if the computers are patched, 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.

3.2.3 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.

3.2.4 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 APT's can be resident on a isolated network (e.g. 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. Honeypots do not detect all malicious attacks so the attacker can stay unnoticed and can still provide information as feedback. Furthermore, the case where the moves of the defender are non stealthy has already been investigated by Laszka [11].

3.2.5 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.

3.2.6 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 it may be disadvantageous if he wants to stay undetected. If the attacker wants to have feedback, the malware needs to make a connection with the attacker to provide this feedback. For example, this can cause more network traffic for the defender to detect.

Periodic strategy

The choice of periodic strategy is motivated by the assumption that in most organisations, 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 on the Second Tuesday of each month. [14]

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 [11], 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.

3.3 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.

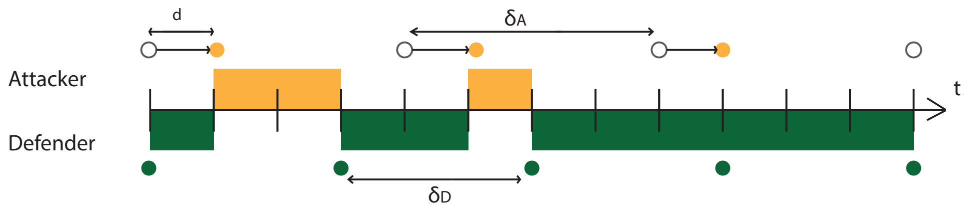


Figure 3.1: Formalization of a FlipIt game with delay: A representation of a FlipIt game where both players are playing periodically. Every move or flip is indicated by a green or orange circle, respectively dark gray and light grey. The defender is represented in green (dark grey) and plays with a period 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 (light grey). 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 form 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 point 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.

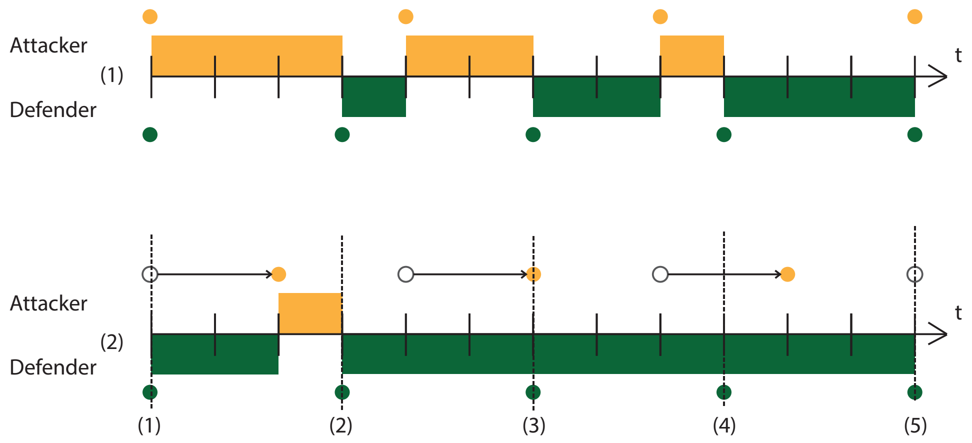


Figure 3.2: The first game is the basic FlipIt game. The second is the FlipIt game with a delay. During the first flip of the attacker, the defender moves after the delay, causing the attacker to get control over the resource. During the second flip of the attacker, the defender flips at time t+d, causing the defender to take control over the resource before the attacker. 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.

Similarly as in [24], we split the formalization in two cases. In the first case the defender plays at least as fast as the attacker, in the second case 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.

Intuitively, we could assume that d can never be bigger than δ_A because then 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 that the period of the defender, then the defender will always be in control. 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. From this we can conclude that it only makes sense to calculate the formulas for the cases where d is smaller than δ_D. We can already conclude that it is of no use for the attacker to play when the delay is bigger than δ_D.

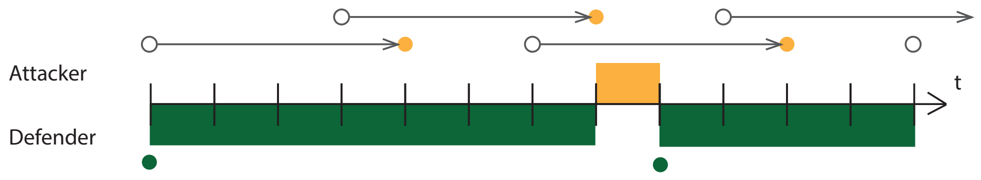


Figure 3.3: FlipIt with delay propagation where $\delta_D > d > \delta_A$.

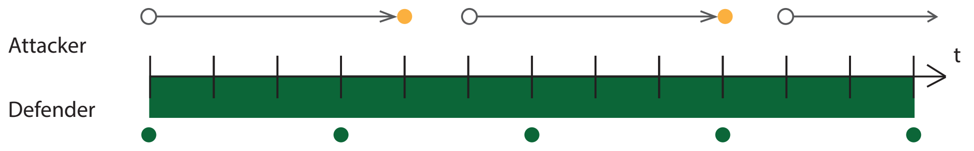


Figure 3.4: FlipIt with delay propagation where $d > \delta_D$

In the original model of FlipIt [24], 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.

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

Let $r = \dfrac{\delta_D}{\delta_A}$. 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:

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

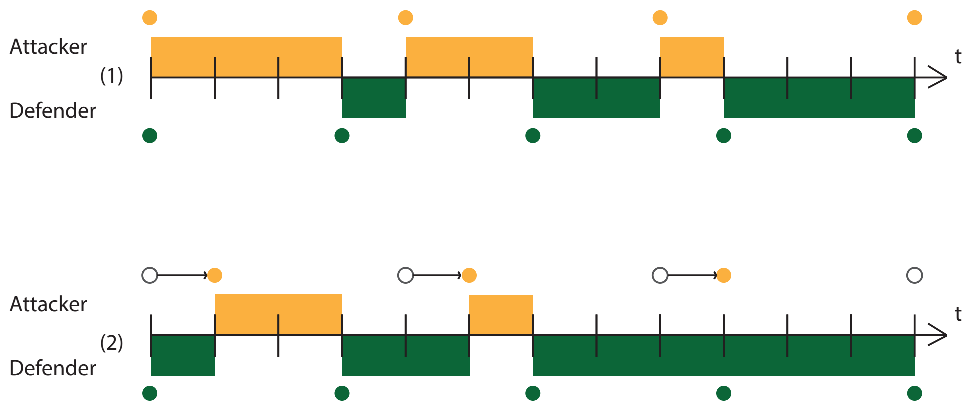


Figure 3.5: Case 1: Difference between a basic FlipIt game and a FlipIt game with a delay. Case (1) is the FlipIt game without a propagation delay and case (2) is with a propagation delay. The delay is denoted with an arrow. The attacker is only in control when the circle becomes orange (light grey).

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 $\dfrac{\delta_D - d}{\delta_D}$, giving the attacker an average gain of $\dfrac{\delta_D - d}{2}$ (the average remainder of the defender interval after the attacker flipped). If the attacker moves

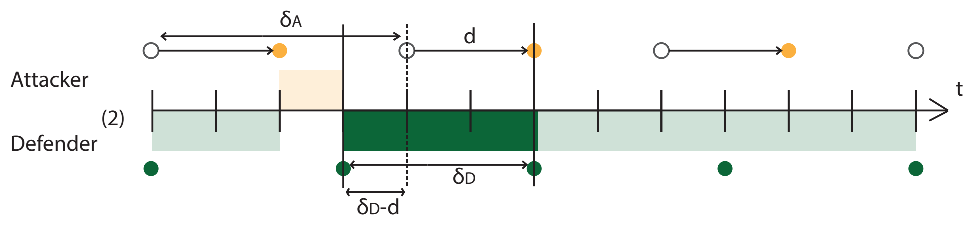


Figure 3.6: Attacker playing to late. If the attacker enters the defender's interval after $\delta_D - d$, he can not get in control in that interval.

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.

$$\beta_A(\delta_D, \delta_A) = \frac{(\delta_D - d)^2}{2\delta_D\delta_A} - \frac{k_A}{\delta_A} \tag{3.1}$$

The benefit of the defender is then as follows:

$$\beta_D(\delta_D, \delta_A) = 1 - \frac{(\delta_D - d)^2}{2 \cdot \delta_D\delta_A} - \frac{k_D}{\delta_D} \tag{3.2}$$

For this case where d=0, we this equals the formula of the original FlipIt game [24] [p675].

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 continuous to be in control during the period of the delay, see figure 3.7. 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 slower 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 a extra gain of d, yielding a total average gain per interval of $\frac{(d + \frac{\delta_A}{2})}{\delta_A}$

For the case with a delay 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.

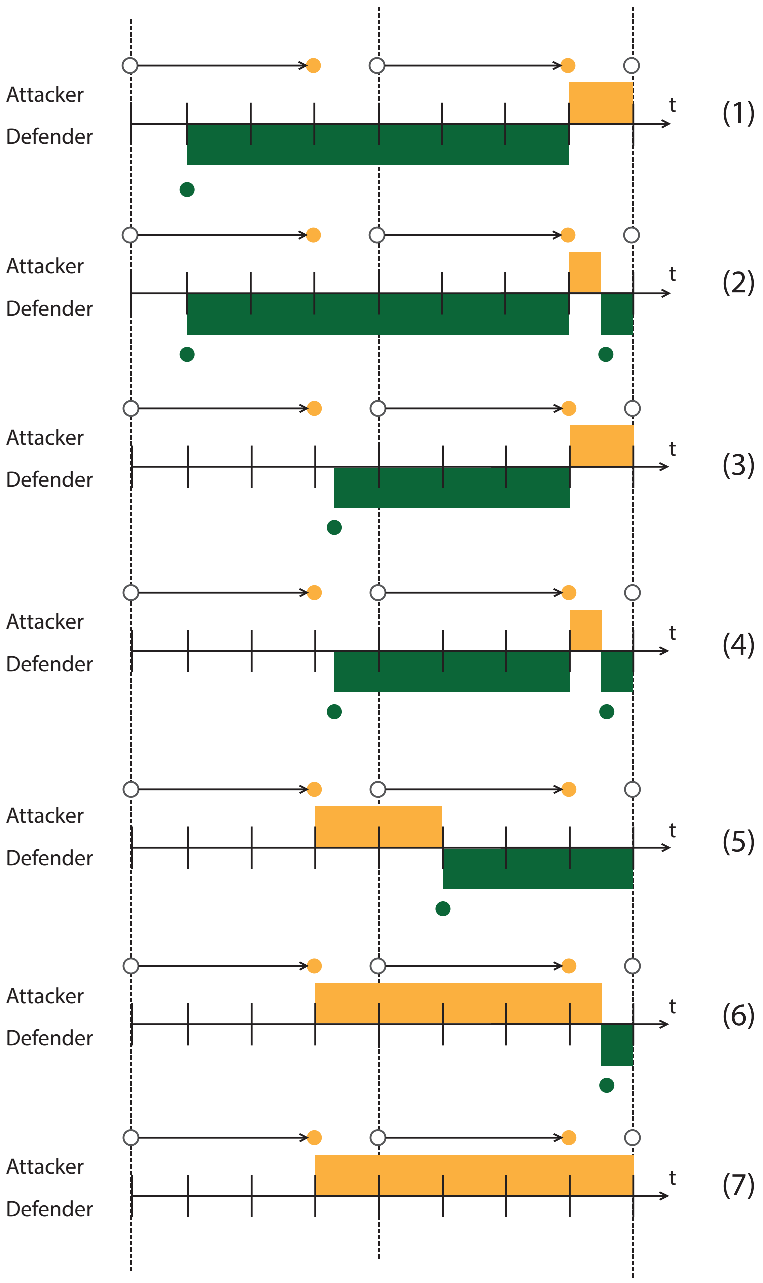


Figure 3.7: All possible cases for the attacker and the defender in Case 2.A where $d+\delta_A < \delta_D$. As can be seen in cases (1) to (4), the defender will have control during a period of d over the resource in the next interval when the defender has flipped in the previous interval.

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:

$$\beta_D(\delta_D, \delta_A) = \frac{\delta_A}{2\delta_D} + \frac{d}{\delta_D} - \frac{k_D}{\delta_D} \tag{3.3}$$

The benefit for the attacker will be as follows:

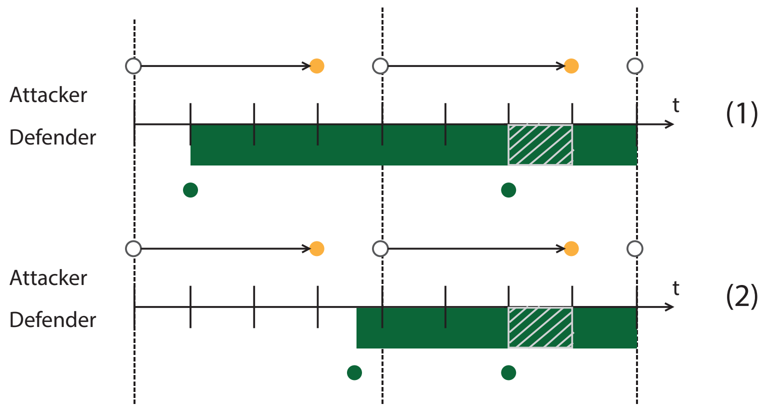
$$\beta_A(\delta_D, \delta_A) = 1 - \frac{\delta_A}{2\delta_D} - \frac{d}{\delta_D} - \frac{k_A}{\delta_A} \tag{3.4}$$

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 to much gain for the defender: the potential overlap during the delay would be counted twice. See figure 3.8

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

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



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 $\dfrac{d-(\delta_D-\delta_A)}{\delta_D}$ and the average gain will be half of this interval: $\dfrac{d-(\delta_D-\delta_A)}{2}$.
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:

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

Consequently, the benefit for the attacker will be:

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

Chapter 4

Nash Equilibria

4.1 Nash Equilibria

In this chapter we are interested in finding the optimal strategies. To calculate the optimal strategies of both players we have to find the Nash equilibria of the game. First the optimal functions are derived from the formulas in the previous chapter. From these piecewise functions we can derive the Nash equilibria.

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 maximized at $\delta_D = \delta_D^*$ and the attacker's benefit $\beta_A(\delta_D^*, \delta_A)$ is maximized 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 optimize 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 optimize the benefit of the attacker for a fixed rate of play δ_D of the defender.

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 two cases, where case 2 is divided into two subcategories.

4.1.1 Determining the piecewise functions $opt_D(\delta_A)$

Case 1: $\delta_D \leq \delta_A$

The benefit formula obtained in the previous chapter (formula 3.2) for the defender in this case is as follows:

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

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:

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

This leads to the following deduction given the sign of the coefficient of δ_D^2: 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 $\delta_D = min\{\delta_A, \sqrt{2\delta_A k_D + d^2}\}$. Taking the minimum of the two values is needed because δ_D cannot be larger than δ_A.

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

The benefit formula obtained in the previous chapter (formula 3.3) for the defender in this case is as follows:

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

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 be always positive but decreasing, see figure 4.1. 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. There is an edge case if $d = k_D$, which results in a benefit of $\beta_D(\delta_D, \delta_A) = \dfrac{\delta_A}{\delta_D}$.

For $\delta_A + 2(d - k_D) < 0$, the benefit will always be increasing but negative so the defender will not play. See figure 4.2.

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

The benefit formula obtained in the previous chapter (formula 3.5) for the defender in this case is as follows:

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

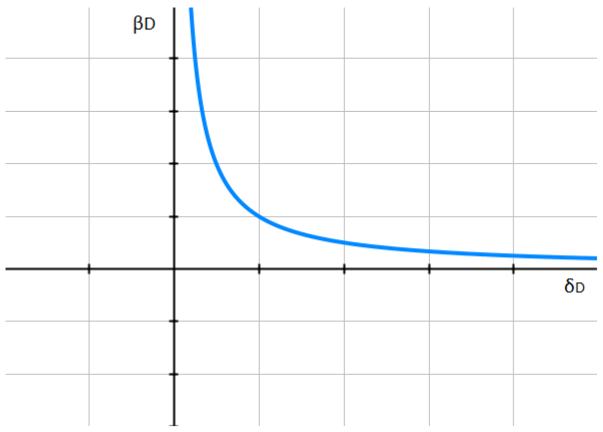


Figure 4.1: The benefit function is of the shape of 1/x if δ_D *is always decreasing and if* $\delta_A + 2(d - k_D) > 0$.

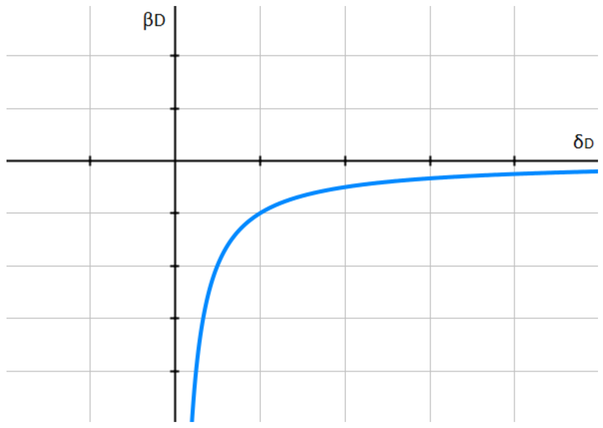


Figure 4.2: The benefit function is of the shape of -1/x if δ_D *is always increasing and if* $\delta_A + 2(d - k_D) < 0$.

The derivative of the above formula for a fixed δ_A results in the following:

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

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

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

For case 2.B this leads to the following deduction, which results in the same formula as for case 1 but with a small difference for the value of δ_D: The function increases on $[0, \sqrt{2\delta_A k_D + d^2}]$ and is decreasing on $[\sqrt{2\delta_A k_D + d^2}, \infty]$. Because $\delta_D \geq \delta_A$ there is a maximum on $\delta_D = maximum\{\delta_A, \sqrt{2\delta_A k_D + d^2}\}$ or because $d + \delta_A \geq \delta_D$ there is also on $\delta_D = minimum\{\delta_A + d, \sqrt{2\delta_A k_D + d^2}\}$.

Best responses

The optimum functions will be piecewise functions. We distinguish three cases for different values of δ_A depending on the values of k_D and d. We point out that δ_A and δ_D are positive rates.

$\delta_A < 2(k_D - d)$

From case 1 above, it follows that the benefit is increasing but it will always be non-positive. The defender will try not to play ($\delta_D = \infty$). From case 2.a and case 2.b it follows that from interval $[2(k_D - d), \sqrt{2k_D\delta_A + d^2}]$ (because for case 2 $\delta_D \geq \delta_A$), that the benefit is increasing. The maximum of case 2.a and the maximum of case 2.b can be brought together. It follows that the optimal benefit is achieved at the $\delta_D = minimum[\delta_A + d, \sqrt{2k_D\delta_A + d^2}]$, which is $\delta_D = \sqrt{2k_D\delta_A + d^2}$. But because of case 2.a, the defender's maximum benefit is not to play at all.

$\delta_A = 2(k_D - d)$

Then for case 2.a, $\beta_D(\delta_D, \delta_A) = 0$ for all $\delta_D \in [\sqrt{2k_D\delta_A + d^2}, \infty]$. For case 2.b and case 1, the benefit is increasing in the interval $[\delta_A, \sqrt{2k_D\delta_A + d^2}]$. If we put both cases back together, we get a maximum at $\sqrt{2k_D\delta_A + d^2}$. So in this case the defender's maximum benefit is achieved for any δ_D in $[\sqrt{2k_D\delta_A + d^2}, \infty]$ with value 0.

$\delta_A > 2(k_D - d)$

From case 2.a above, it follows that the benefit is decreasing but positive. So the defender will try to play as fast as possible ($\delta_D \to 0$). For case 2.b it is increasing in the interval $[\delta_A, \sqrt{2k_D\delta_A + d^2}]$ and for case 1 it is increasing in the interval $[0, \delta_A]$. It follows that the defender's maximum benefit is achieved for $\delta_D = \sqrt{2k_D\delta_A + d^2}$

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

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

4.1.2 Determining the piecewise functions $opt_A(\delta_D)$

To start with we only consider the case where $d < \delta_D$, because if $d > \delta_D$ the benefit of the defender is always 1.

nakijken, def of att

To determine the Nash equilibria 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 three cases:

Case 1: $\delta_A \geq \delta_D$

The benefit formula obtained in the previous chapter 3.1 for this case is as follows:

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

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:

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

It follows that $\beta_A(\delta_D, \cdot)$ is increasing and non-positive if $\delta_D > \frac{(\delta_D - d)^2}{2k_A}$ and decreasing and positive if $\delta_D < \frac{(\delta_D - d)^2}{2k_A}$

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

The benefit formula obtained from the previous chapter 3.4 for this case is as follows:

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

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 $\delta_A = \mathit{minimum}\{\delta_D - d, \sqrt{2k_A\delta_D}\}$. 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 2.B: $d + \delta_A \geq \delta_D \geq \delta_A$

The benefit formula obtained from the previous chapter 3.6 for this case is as follows:

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

The derivative for a fixed δ_D is as follows:

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

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

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

It follows that $\beta_A(\delta_D, \cdot)$ is increasing and non-positif if $\delta_D > \frac{(\delta_D - d)^2}{2k_A}$ and decreasing and positive if $\delta_D < \frac{(\delta_D - d)^2}{2k_A}$. This is the same result as in case 1.

Best responses

The optimum functions will be piecewise functions. We distinguish three cases for different values of δ_D regarding d and k_A.

For this term $\frac{(\delta_D - d)^2}{2k_A}$, d has to be bigger than δ_D because the cost δ_D cannot be negative. This was an assumption that was already made, because the benefit of the defender will always be 1 if d is bigger than δ_D.

$\delta_D < \frac{(\delta_D - d)^2}{2k_A}$

From case 1 and case 2.b above, it follows that the benefit is decreasing but positive, which means that the attacker will play as fast as he can. For case 2.a $\delta_D \geq \delta_A$ is increasing. It follows that for this case the attacker's maximum benefit is achieved for $\delta_A = \sqrt{dk_A\delta_D}$. . In this case the attacker's maximum benefit is reached at $\delta_A = \delta_A = \sqrt{dk_A\delta_D}$.

$\delta_D = \frac{(\delta_D - d)^2}{2k_A}$

The benefit $\beta_A(\delta_D, \delta_A) = 0$ for case 1 and case 2.b for all δ_D in $[0, \infty[$. For case 2.a it follows that the benefit is increasing. The attacker's maximum benefit is achieved at any δ_A in

$\delta_D > \dfrac{(\delta_D - d)^2}{2k_A}$

For case 1 and case 2.b the benefit is increasing but non-positive. This means that the attacker won't play ($\delta_A = \infty$).

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

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

The Nash Equilibria can be found as the intersection points of the piecewise functions $opt_A(\delta_D)$ and $opt_D(\delta_A)$. For this we have to compare the function in terms of the relationship of the players move cost. We distinguish three cases: $k_A < k_D$, $k_A > k_D$ and $k_A = k_D$.

kA met kD vergelijken en nash evenwichten vinden

een voorbeeld invullen met specifieke waarden

Chapter 5

Models for the Delay

The formalisation 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 itself.
Viruses and worms are the types of malware that are the most researched. 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 kinds of models in section 5.2 illustrates a couple of models as examples to determine the propagation of a worm. These models can be used to extract parameter d. Section 5.3 introduces an easy method to calculate the delay of the propagation of a worm. In the last section 5.3, a method based on the PageRank algorithm of Google is used to determine which node has a higher probability of being infected by a worm.

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 fishing or trojan horses. To determine the total delay, however, the propagation strategy is of higher importance than the drop mechanism. Propagation is achieved by determining the next nodes to spread the worm 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 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* [23] and *Slammer* [15].

Voorbeeeld als mensen het idee niet snappen?

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* [23] and *Nimda worm* [23].

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* [6].

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*.

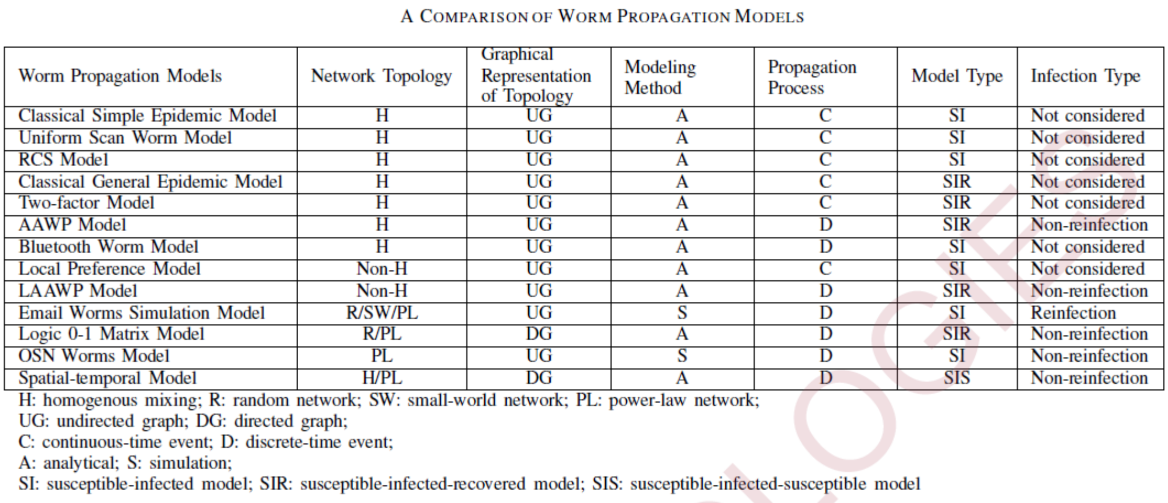
Topology-based Worms Email and other client application worms:An email worms uses the email systems to find email addresses to propagate. Other client applications can include: Internet Relay Chat (IRC), Instant Messenger (IM), and a variety of peer-to-peer file sharing systems, which have been used by worms to propagate in a similar way as email worms. For example, the *Kak worm* [20] 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, SIR and SIS. 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 ([26], [18], [28] and [21]). We are only interested in models that will allow us to find the delay.

Table 5.1 gives an overview of the most common propagation methods with the given type. This table comes from the most recent survey [26], which encompasses results of older surveys. Moreover, the survey explicitly focuses on propagation methods, while other papers also focus on other aspects such as detection mechanisms and containment systems. All models that are based on SI model type (Susceptible-Infected) are good, because in our model every node that is infected will stay infected. The other propagation models, SIR and SIS, are only interesting if there is a possibility to reduce the models to an SI model. The following section will apply the extraction of the delay to a few models. These models are not meant as an



A Comparison of Worm Propagation Models

Worm Propagation Models	Network Topology	Graphical Representation of Topology	Modeling Method	Propagation Process	Model Type	Infection Type
Classical Simple Epidemic Model	H	UG	A	C	SI	Not considered
Uniform Scan Worm Model	H	UG	A	C	SI	Not considered
RCS Model	H	UG	A	C	SI	Not considered
Classical General Epidemic Model	H	UG	A	C	SIR	Not considered
Two-factor Model	H	UG	A	C	SIR	Not considered
AAWP Model	H	UG	A	D	SIR	Non-reinfection
Bluetooth Worm Model	H	UG	A	D	SI	Not considered
Local Preference Model	Non-H	UG	A	C	SI	Not considered
LAAWP Model	Non-H	UG	A	D	SIR	Non-reinfection
Email Worms Simulation Model	R/SW/PL	UG	S	D	SI	Reinfection
Logic 0-1 Matrix Model	R/PL	DG	A	D	SIR	Non-reinfection
OSN Worms Model	PL	UG	S	D	SI	Non-reinfection
Spatial-temporal Model	H/PL	DG	A	D	SIS	Non-reinfection

H: homogenous mixing; R: random network; SW: small-world network; PL: power-law network;
UG: undirected graph; DG: directed graph;
C: continuous-time event; D: discrete-time event;
A: analytical; S: simulation;
SI: susceptible-infected model; SIR: susceptible-infected-recovered model; SIS: susceptible-infected-susceptible model

Figure 5.1: Taxonomy of worm modelling. Image based on taxonomy given in [26]

.

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 amount of nodes in the network. If formula 5.2 is equal to this amount, variable t will be the value of the delay that we need. This is only applicable to nodes in a homogeneous network. A homogeneous network is a network where every node has approximately the same degree. 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 [23]. Another drawback 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

RCS model stands for Random Constant Spread model and is developed by Paxson and Weaver at Stanford [23]. It is a model derived from the classical Simple Epidemic Model. They used this model 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, which means that the number of hosts in the system stay constant. The model also ignores spreads of the worm behind firewalls on private networks.
Let K be the initial compromise rate. This is the rate of hosts that the worm can find and compromise per hour 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. It cannot increase the rate that a worm can find new hosts. The internet topology is considered as a complete undirected graph. The model type is SI, so the state of the host can only be infected or suspected. Once the host is compromised it stays that way. 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 it is known the proportion of machines a that already have been compromised.

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 a variable in the above formula that says how many time has passed. 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 for this model is that the network topology is homogeneous and that the graphical representation of the graph is an undirected graph.

5.2.3 Bluetooth worm model

The Bluetooth worm model is 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*:

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

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).

The formula for a better estimation of the worm propagation:

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

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.

5.2.4 AAWP model

AAWP stands for Analytic Active Worm Propagation. The model is proposed in [3] 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 hosts cannot infect another hosts before it is completely compromised. The model also considers the fact that a hosts 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. To extract the delay from this formula we need to know when the iteration procedure stops.
The biggest drawback is that the model is a discrete time model. Continuous models are more appropriate for large scale models. The model also uses a complete undirected graph.

5.3 Matrix worm model

For some of the propagation methods, the graph of the network matters. It is important to have the right topology for the right method. Email worms need a topology that represent a social network, BGP routing worms need a topology on network level.

As can be seen in table 5.1 all propagation models are depending on a specific network topology. The different kinds of network topology of each system is either homogeneous, small-world network, random network or a power-law network. A homogeneous network is a network where every node has about the same degree of connectivity. This means that every node can infect each node with the same opportunity. A small-world topology is a network where most nodes are not connected with each other but they can all reach each other in a few steps. Social networks are an example of a small-world network. A random network is a topology where 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.

What we want to have is a method to calculate the delay that is independent of a network topology. A method where we can chose in the beginning which topology

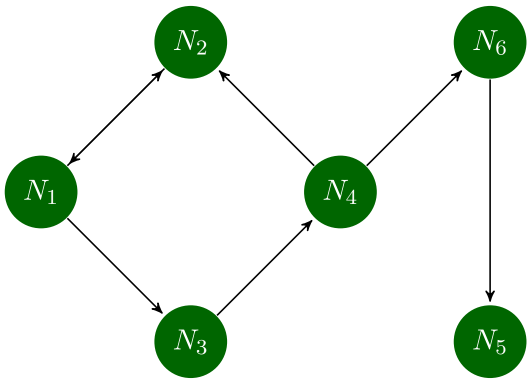


Figure 5.2: Network with 6 nodes. The arrows represent the connections between the nodes.

we have and then calculate the delay.

In this chapter we propose a method to calculate the spread of a worm in a way that the topology of the network can be easily integrated. The method will give an approximation of how fast a worm can infect a network. This method is capable of calculating the delay for any network topology, but it is not the most efficient way of doing this. For this paper it is just a proof of concept.

Model with matrix

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 where 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:

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

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 that indicates which nodes are infected after a 3 time steps:

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

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

Conclusions

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.
With this method it is also possible to determine the shortest path in the network. By summing up all the matrices, the first row that has all non-zero entries determines the shortest 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.
A disadvantage of this method is the cost. A very large network needs a very large (sparse) matrix.

5.4 Google PageRank algorithm

To calculate the delay of a worm propagation we assumed that every node is even important. A possible relaxation of this rule can be done by the PageRank Algorithm, which can be used to determine the importance of certain nodes.

The PageRank algorithm is 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 is 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 will point to this page. This means that this page is well cited through other pages and maybe an important one 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 depends than on the number of out coming 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:

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

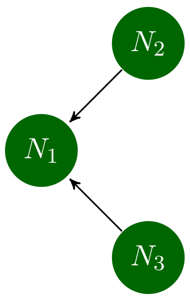


Figure 5.3: A representation of a graph with three nodes. Node 1 is a dangling node with no outgoing links.

where N_A is the number of pages, $PR(i)$ is the ranking 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 Google matrix is a stochastic matrix that is used to calculate the PageRank of all the web pages on the internet. The dominant eigenvalue of the matrix is the PageRank. The matrix is a graph where the edges denote the links between each page. Using the power method, the PageRank of each page can be calculated iteratively.

The Google Matrix of a graph is defined as follow:

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

where p defines the damping factor and indicates the probability that a surfer quits a current page and continuous on a new random page. Matrix A is an transposed matrix of the graph matrix. The graph matrix is 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 link from one page to the other page. For dangling nodes (nodes with no outgoing links), the row entries are equal to $\frac{1}{n}$. Matrix A for the graph represented figure 5.3 is equal to:

$$A = \begin{array}{c} \\ N_1 \\ N_2 \\ N_3 \end{array}\begin{array}{c} \begin{array}{ccc} N_1 & N_2 & N_3 \end{array} \\ \begin{pmatrix} 1/3 & 1 & 1 \\ 1/3 & 0 & 0 \\ 1/3 & 0 & 0 \end{pmatrix}\end{array}$$

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 Method.

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 op time. When a worm uses the topology of the graph to propagate the damping variable p must be equal to zero, as in the worm will not suddenly attack a node that is not connected. When the propagation of a worm is topology independent, the damping variable p can be set on 1.
This can be useful for further research, when the defender can also flip a subset of nodes. It can be advantageous to flip the nodes with a higher PageRank more often than the other nodes. A higher PageRank means that the page has a bigger probability of being compromised by a worm.

Chapter 6

Conclusion

This paper presents an adaption to the basic FlipIt game by [24] to model an attacker with a delay. We have

The work presented in this paper represents an initial caclulation of the strategies.

6.0.1 Further work

Analysing other renewal strategies.

Delay for the defender. It takes some time to make a patch if a new exploit is found.

The delay a variable that can change with every Flip.

The defender can flip a subset of nodes instead of all the nodes.

6.1 trala

Bibliography

[1] J. Aspnes, K. Chang, and A. Yampolskiy. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 43–52. Society for Industrial and Applied Mathematics, 2005.

[2] 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.

[3] 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.

[4] Cornell. Google page rank, 2009.

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

[6] 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.

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

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

[9] 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.

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

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

[12] 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.

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

[14] Microsoft. security bulletin release.

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

[16] 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.

[17] 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.

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

[19] 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.

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

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

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

[23] 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.

[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 that moreover such attacks can go undetected for a while. These kind of attacks are called APT, advanced persistent threats, which are designed to penetrate secretly a computer network, collect sensitive data and stay hidden for many years. A company has every interest to mitigate the risks of an APT and the consequences that it can cause. Because of the stealthiness fighting against these attacks 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 of controlling the resource and minimise the total move cost.
FlipIt does however not take into account that a move may not be instantaneous, but has 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 a virus. The FlipIt formulas are adapted such as to take the delay for virus 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 thesis is to find out if their are interesting Nash Equilibria for a game with a virus propagation delay and if we can learn some lessons out of it.

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