

Flip the virus: Modelling targeted attacks using Flipt 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

© Copyright KU Leuven

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.

Voorwoord

Thank you !

Sophie Marien

Inhoudsopgave

Voorwoord	i
Samenvatting	iv
Samenvatting	v
Lijst van figuren en tabellen	vi
List of Abbreviations and Symbols	viii
1 Introduction	1
1.1 Introduction	1
2 General context	5
2.1 What is security?	5
2.2 A brief introduction in Game Theory	8
2.3 The FlipIt game	10
2.4 Related Work on Extensions to FlipIt	13
3 FlipIt with propagation delay	17
3.1 Introduction	17
3.2 Difference between FlipIt with and without propagation delay	17
3.3 Formalization of the periodic game with propagation delay	20
4 Nash Equilibria	29
4.1 Nash Equilibria	29
5 Models for the Delay	37
5.1 Methods of propagation	37
5.2 Models for worm propagation	39
5.3 methode met matrixen	43
6 Conclusion	47
6.1 trala	47
Bibliografie	49

Todo list

Results toevoegen	iv
en het wordt ook toegepast op de formules	3
niet volledig juist, defender systemen geplaatst en dan gezien hoe ver de worm kan geraken	18
formule nog eens nakijken	25
dat laatste nog eens nakijken	33
juiste resultaten hier nog invullen	35
kA met kD vergelijken en nash evenwichten vinden	35
een voorbeeld invullen met specifieke waarden	35

Samenvatting

Recently, high profile targeted attacks such as the attack on Belgacom (a major Belgian telcom), have demonstrated that even the most secure companies can still be compromised, and that moreover such attacks can go undetected for a while. These kind of attacks are called APT, advanced persistent threats and are designed to secretly penetrate 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 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 thesis is to find out if modelling such situations with FlipIt with propagation delay allows us to draw interesting lessons about security measures against APT’s.

Results toevoegen

Keywords: Game theory, Advanced persistent threats, cyber security, FlipIt, stealthy takeovers, propagation methods.

Samenvatting

Nederlandse abstract

Lijst van figuren en tabellen

Lijst van figuren

2.1	Prisoners dilemma: An example from the field of Game Theory. It is a game between two perfectly rational prisoners who don not know what the other one will do. Each can confess or stay silent. The resulting sentence is shown below each prisoner.	10
2.2	A representation of a FlipIt game where both players are playing periodically. Every move or flip is indicated by a blue or orange circle. The attacker is represented in orange and the defender is represented in blue. The blue and orange rectangles represent the amount of time the respective player is in control of the resource.	11
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 blue or orange circle, respectively dark gray and light grey. The defender is represented in blue (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 blue and orange rectangles represent the amount of time the respective player is in control of the resource.	20
3.2	The first game is the basic FlipIt game. The second is a 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.	21
3.3	FlipIt with delay propagation where $\delta_D > d > \delta_A$	22
3.4	FlipIt with delay propagation where $d > \delta_D$	22
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).	23

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.	24
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.	26
3.8	Cases where the delay would be counted twice.	28
4.1	This figure shows the three subcategories of each case. 'A' stands for Case 2.A: $\delta_D \geq d + \delta_A \geq \delta_A$ and 'B' stands for Case 2.B: $d + \delta_A \geq \delta_D \geq \delta_A$	30
4.2	The benefit function is of the shape of $1/x$ if δ_D is always positive and if $\delta_A + 2(d - k_D) > 0$	31
4.3	The benefit function is of the shape of $-1/x$ if δ_D is always positive and if $\delta_A + 2(d - k_D) < 0$	31
5.1	Taxonomy of worm modelling. Image based on taxonomy given in [21] .	40
5.2	Simple Epidemic Model. The x-coordinate is the propagation time and the y-coordinate is the infected percentage of the whole network. Original image from [15]	41
5.3	KM Model. The x-coordinate is the propagation time and the y-coordinate is the infected percentage of the whole network. Original image from [15]. $N = 10000, \beta = 1/10000000$	42
5.4	Network with 6 nodes. The arrows represent the connections between the nodes. The start point is were the worm has been dropped, here node 1.	44

Lijst van tabellen

2.1	Hierarchy of Classes of strategies in FlipIt	13
-----	--	----

List of Abbreviations and Symbols

Abbreviations

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
SIR	Susceptible - Infected - Recovered
SIS	Susceptible - Infected - Susceptible

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.

Hoofdstuk 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 on clients, 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 results 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 stronger 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 themselves against APT's: awareness, whitelisting, system administration, network segregation, dynamic content checking and patch management. Nevertheless the combination of all these elements benefit from being complemented by other defence strategies to protect one selves 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 [4]. 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 taking defensive measures that protect the system.

Many security games that bridge 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” [20]. 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 sufficient 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 [20] 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).

en het wordt
ook toegepast
op de formules

While it may seem trivial to extend the basic FlipIt model with a propagation delay, it's mathematical treatment is not. In the paper of Laznka et al. [7] it seems that even a small extension adds to the already significant mathematical complexity. While the FlipIt game is quite symmetric, by adding a propagation delay the mathematical complexity rises.

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 get 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 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, we discuss the main results and provides directions for further research.

Hoofdstuk 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 for 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: the game that 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.

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

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 sub categories. 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. The 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 the 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

In order to keep a system secure and to be able to apply security tools, it is important to mitigate the possible threats. 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 [5] reveals that 61% of the attacks on companies are caused by malware. For this reason this section will examine the categories of malware threats. Different types of malware exist: viruses, worms, flooders, rootkits, bots, spyware, adware and many more. This broad range of different types can be classified into two main categories, the first one based on

the propagation method that is used and the second one based on the payload or the variety of actions that the malware performs [18]. 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 an 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 could not 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 worm sent a mail or friend request with a message that directed the recipients to a third party website. This site would then convince the recipient into downloading an update of Adobe Flash Player. Once downloaded and executed, Koobface could infect the host.

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

The known threats are the easiest to defend oneself against. Standard malware protection tools like firewalls and virus scanners can keep these kind of malware

out of the system. Installing protection against unknown threats is also relatively easy, but this requires tools that go beyond the standard methods, e.g. dynamic whitelisting. The remaining 1% are the advanced threats, also known as APT. They are the most difficult to deal with.

Advanced threats to computer systems

An Advanced Persistent Threat (referred to in the rest 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. APT's are 'advanced' because these attacks are well funded and (usually) the attacker needs a great amount of expertise to successfully penetrate a network. Not all APT's are *technically* advanced though. The attacker can also try to exploit known vulnerabilities, in the hope that his target has not yet secured himself against them. 'Persistent' refers to the fact that the attacker never quits trying to attack his target. 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 like the way a virus, worm or trojan horse propagates.

According to a security survey of Kaspersky [5] the damage of one successful targeted attack against a large company can exceed over 2.54 million dollar. Companies need a defence mechanism to defend themselves against APT's. As previously stated, simple detection and identification tools are insufficient to protect oneself against APT's, and a removal tool will only work if the threat has been identified. To mitigate these kind of attacks another security countermeasure is needed.

In this paper we will propose an appropriate game theoretical modelling of defending a network against APTs and analyse it in order to draw the necessary strategic conclusions to mitigate these kind of attacks.

2.2 A brief introduction in Game Theory

Gametheory is a mathematical study to analyse interactions between independent and self-interested agents. To get an understanding of the most important concepts of game theory, a short introduction based on the work of [10] and [11] is given in this section. For a more detailed and full introduction to game theory, the reader is referred to [10].

Game theory is a mathematical way of modelling the interactions between two or more agents where the outcomes depend on what everybody does and how it should be structured to lead to good outcomes. Game theory has therefore important applications in many area's such as economics, politics, biology, computer science, philosophy and a variety of other disciplines. It has come only in ascent during the Second World War with Oskar Morgenstern and John von Neumann both publishing a book on game theory, entitled "Theory of Games and Economic Behavior"(Game theory and economic behaviour) in 1944. This book was about the mathematical

analysis of a series of thinking games. The book described a series of games making a distinction between games in which the strategies and the utility factor of the opponent have no effect on finding the best strategies such (e.g. chess) and games in which this has 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 was named after him.

One of the assumptions underlying game theory is that the players of the game, the agents, 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 the states of the world he likes. These preferences are mapped to natural numbers and are called the utility function. The numbers are interpreted as a mathematical measure that tells how much an agent likes or dislikes the states of the world.

In a decision game theoretic approach an agent will try to act in such a way to maximise his expected or average utility function. It becomes more complicated when two or more agents want to maximise their utility and when actions of the agents can affect each other's utilities. This kind of games are referred to as non-cooperative game theory, where the basic modelling unit is the group of agents. The individualistic approach, where the basic modelling is only one agent, is referred to as cooperative game theory.

Best response and Nash Equilibrium

One of the solution concepts in Game Theory for non-cooperative games is a Nash Equilibrium that will be used in this paper. 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 for a player is the action of a player that maximizes its pay-off for any given action of the other player. 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 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 confesses, he will, depending on what the other does, go for three years in prison or he is free to go. If the player is silent, he will, depending on what the other

2. GENERAL CONTEXT

player does, go for five years in jail or only one year.

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 confess. 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 confess even though it would be better if both prisoners cooperated and choose to stay silent.





		Prisoner 2	
		Talk	Stay Silent
Prisoner 1	Talk	 3 YEARS 3 YEARS	 FREE 5 YEARS
	Stay Silent	 5 YEARS FREE	 1 YEAR 1 YEAR

Figure 2.1: Prisoners dilemma: An example from the field of Game Theory. It is a game between two perfectly rational prisoners who don't know what the other one will do. Each can confess or stay 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 ($t \rightarrow \infty$). 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 to move too frequently.

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

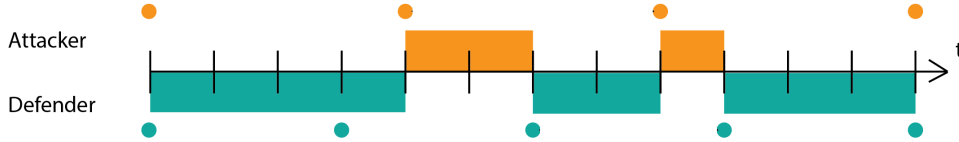


Figure 2.2: A representation of a FlipIt game where both players are playing periodically. Every move or flip is indicated by a blue or orange circle. The attacker is represented in orange and the defender is represented in blue. The blue and orange rectangles represent the amount of time the respective 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 players receive a benefit equal to the time units they were in possession of the resource minus the cost of making their moves. The cost of a player i is denoted by k_i . The total gain of player i is equal to the total amount of time that a player i has owned the resource from the beginning of the game up to time t . It is expressed as follows:

$$G_i(t) = \int_0^t C_i(x) dx. \quad (2.1)$$

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

$$G_D(t) + G_A(t) = t \quad (2.2)$$

The average gain rate of player i is defined as:

$$\gamma_i(t) = G_i(t)/t. \quad (2.3)$$

And thus for all $t > 0$:

$$\gamma_D(t) + \gamma_A(t) = 1 \quad (2.4)$$

α_i defines the average move rate by player i up to time t . Let $\beta_i(t)$ denote player's i average benefit upto time t :

$$\beta_i(t) = \gamma_i(t) - k_i \alpha_i. \quad (2.5)$$

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

$$\beta_i(t) = \lim_{t \rightarrow \infty} \inf \beta_i(t) \quad (2.6)$$

Strategies

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

If there is no feedback for either player, we have a non-adaptive strategy. Because a player does not receive any feedback during the game he will play in the same manner against every opponent. The strategy is called non-adaptive because the playing strategy is not dependent on the opponents movements. An interesting subclass of the non-adaptive strategies is the one where the time intervals between two consecutive moves are generated by a renewal process. An example of such renewal strategy is the periodic strategy where the time between two consecutive moves of the players are a fixed interval. An exponential strategy is a renewal strategy in which the interval between two consecutive moves is exponentially distributed.

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

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

Tabel 2.1: Hierarchy of Classes of strategies in FlipIt

Results of the FlipIt game

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

- periodic strategy strongly dominates the other renewal strategies if the opponent has a periodic or non-arithmetic renewal strategy, meaning that it is always advantageous 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;
- if a defender facing an LM attacker and with a cost much lower than the attacker, he can play with a periodic rate that is fast enough he'll force the attacker to drop out;
- a defender facing an LM attacker has to play with a randomized strategy e.g. exponential strategy;
- any amount of feedback about the opponent received during the game, benefits to a player.

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. An example is that gaining access to deeper layers of a system may require breaking several passwords. The model is called FlipThem [7]. 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 [8] is extending the game to also consider non-targeted attacks by non-strategic players. In this game the defender tries to maintain control over the resource that is subjected to both targeted and non-targeted attacks. Non-targeted attacks can include phishing, while targeted attacks may include threats delivered through zero day attack vulnerabilities. One of the last important additions from Laszka et al. [9] is to consider a game with targeted and non-targeted attacks where the moves made by the attacker do not succeed immediately. This approach is similar to what will be done in this paper, but nevertheless has some major difference. In Laszka's paper, firstly 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 by the use of honeypots. 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. This is a way how the defender can also be stealthy by providing false information through the honeypot. 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 till it succeeds is given by a Poisson process. In our paper the delay is given by one parameter, that can be the result of any virus propagation model. The third and last difference is that the paper of Laszka has multiple attackers and they 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 where required for the FlipIt model. One of the scenarios by Pham [14] 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 [4].

Finally researchers also have investigated the behaviour of humans playing FlipIt. A. Nochenson and Grossklags [13] investigate 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. [16] extended the work of A. Nochenson and Grossklags to include various visual presentation modalities for the available feedback during the investigation.

Hoofdstuk 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].

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.

niet volledig
juist, defender
systemen ge-
plaatst en dan
gezien hoe ver
de worm kan
geraken

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 [9].

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. [12]

In the basic FlipIt game [20] the authors van Dijk et al. concluded that a periodic strategy is the dominant strategy against all other renewal strategies if the opponent uses a periodic or non-arithmetic renewal strategy. So regardless of whether the defender chooses to play periodically or not, it's a good choice for the attacker to play periodically as well.

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

3.3 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 . Moreover it has a random phase, that is chosen uniformly and random in the interval $[0, \delta]$ for the first move. The average rate of play of a player is denoted by $\alpha_i = \frac{1}{\delta_i}$. Stated below a list of symbols that will be used throughout the paper. Figure 3.1 represents the main symbols for clarification.

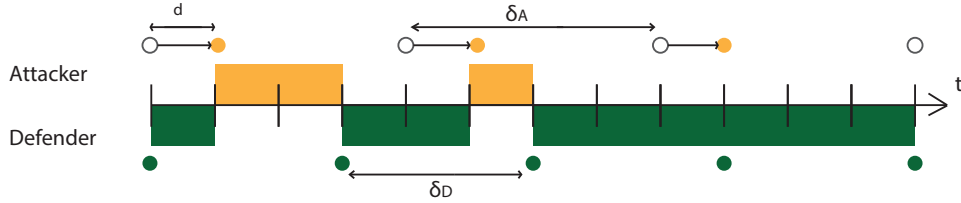


Figure 3.1: Formalization of a FlipIt game with delay: A representation of a FlipIt game where both players are playing periodically. Every move or flip is indicated by a blue or orange circle, respectively dark gray and light grey. The defender is represented in blue (dark grey) and plays with a period 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 blue and orange rectangles represent the amount of time the respective player is in control of the resource.

- i*: Denotes the player. ‘D’ denotes the defender, and ‘A’ denotes the attacker which differs from the notation in [20], 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 its 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 .

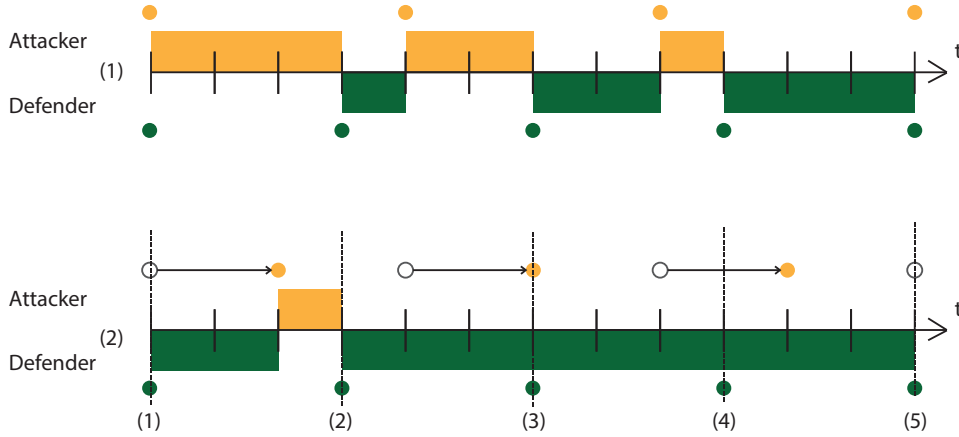


Figure 3.2: The first game is the basic FlipIt game. The second is a FlipIt game with a delay. During the first flip of the attacker, the defender moves after the delay, causing the attacker to get control over the resource. During the second flip of the attacker, the defender flips at time $t+d$, causing 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 [20], we split the formalization in two cases. The first case is where the defender plays at least as fast as the attacker, the second case is where the attacker plays at least as fast as the defender. For each of these cases, first the benefit formula of the basic case without delay is presented, and subsequently the delay is introduced.

Intuitively, we could assume that d can never be bigger than δ_A because then the benefit for the defender would always be 1, but this is not always true. Assume for example that an attacker plays with an interval of 3 units, the delay is equal to 4 units and that the defender only plays every 8 units. This situation is represented in figure 3.3. Since the delay is shorter than the period of the defender, the attacker takes control of the resource once the delay has elapsed, until the defender plays. However, if the delay is larger than 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 units, the delay is equal to 6 units and the defender plays with an interval of 4 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 no use for the attacker to play when the delay is bigger than δ_D .

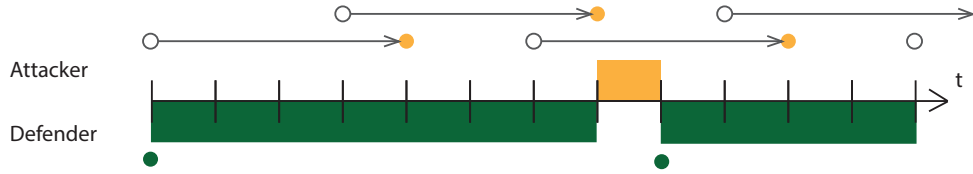


Figure 3.3: FlipIt with delay propagation where $\delta_D > d > \delta_A$.

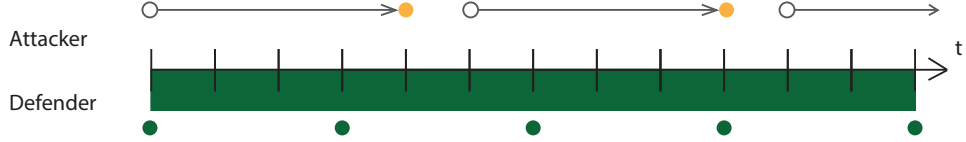


Figure 3.4: FlipIt with delay propagation where $d > \delta_D$

In the original model of FlipIt [20], 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 what result is the simplest formula.

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

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

his move is distributed uniformly at random within this interval.

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

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

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

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

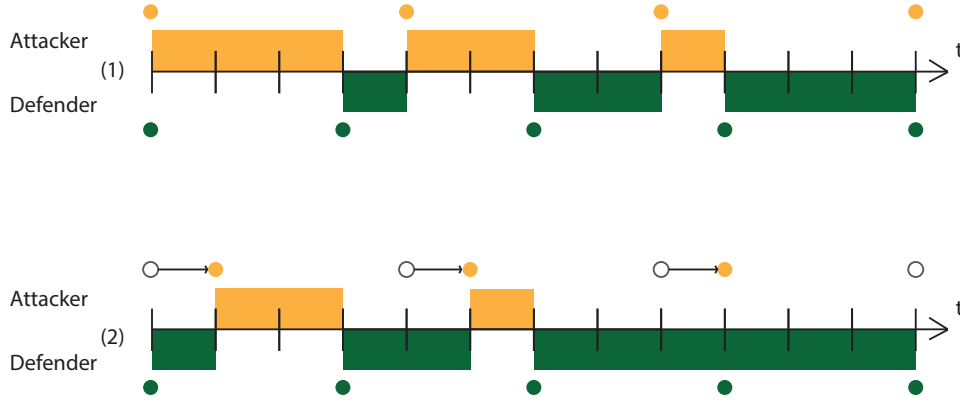


Figure 3.5: Case 1: Difference between a basic FlipIt game and a FlipIt game with a delay. Case (1) is the FlipIt game without a propagation delay and case (2) is with a propagation delay. The delay is denoted with an arrow. The attacker is only in control when the circle becomes orange (light grey).

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 get in control in that interval (see figure 3.6). The probability that the attacker plays soon enough is $\frac{\delta_D - d}{\delta_D}$ and this will give the attacker an average gain of $\frac{\delta_D - d}{2}$. If the attacker moves after the period of $\delta_D - d$, the gain of the attacker will be zero. The probability that this happens is $\frac{d}{\delta_D}$. Looking at one interval of the defender, the average gain rate of the attacker can thus be expressed as follows:

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

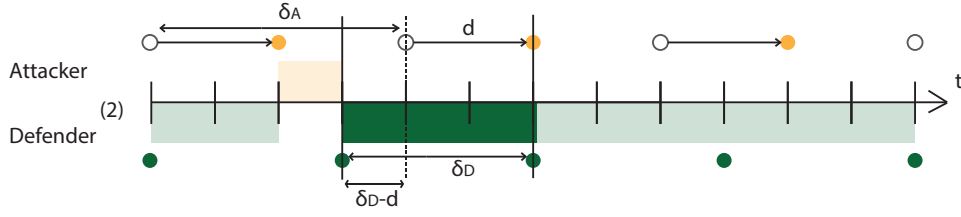


Figure 3.6: Attacker playing to late. If the attacker enters the defender's interval after $\delta_D - d$, he can not get in control in that interval.

Thus over the entire game, the average gain rate is:

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

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

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

The benefit of the defender is then as follows:

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

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

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

First let $r = \frac{\delta_D}{\delta_A}$. The intervals between two consecutive attacker's moves have length δ_A . Consider a given attacker's 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:

$$\begin{aligned}\beta_D(\delta_D, \delta_A) &= \frac{1}{2r} - k_D\alpha_D = \frac{\delta_A}{2\delta_D} - \frac{k_D}{\delta_D} \\ \beta_A(\delta_D, \delta_A) &= 1 - \frac{1}{2r} - k_A\alpha_A = 1 - \frac{\delta_A}{2\delta_D} - \frac{k_A}{\delta_A}\end{aligned}$$

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. By simulation the game, it can be noticed that the closer δ_A/δ_D is equal to one, the better the approximation. If $\delta_A/\delta_D = 1$ the result is correct. The reason this formula overestimates the benefit of the attacker is that it assumes that the defender is always in control during the delay. However, if the attacker was in control in the previous interval, then he continues to be in control during the period of the delay, see figure 3.7. This means that to find the correct formula, what happens in the previous interval needs to be taken into account.

We know that the defender's moves are instantaneous. Therefore, it is easier in this case 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 $\frac{\delta_A}{\delta_D}$. When 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 virus we consider two cases, Case a and Case b, depending on whether the delay is shorter or longer than the difference between the attacker's and the defender's period.

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

The first case is where the defender's period is larger than the sum of the delay and the attacker's period. In that case there is no problem of counting the delay twice. To determine the total gain rate of the defender we need to know what the

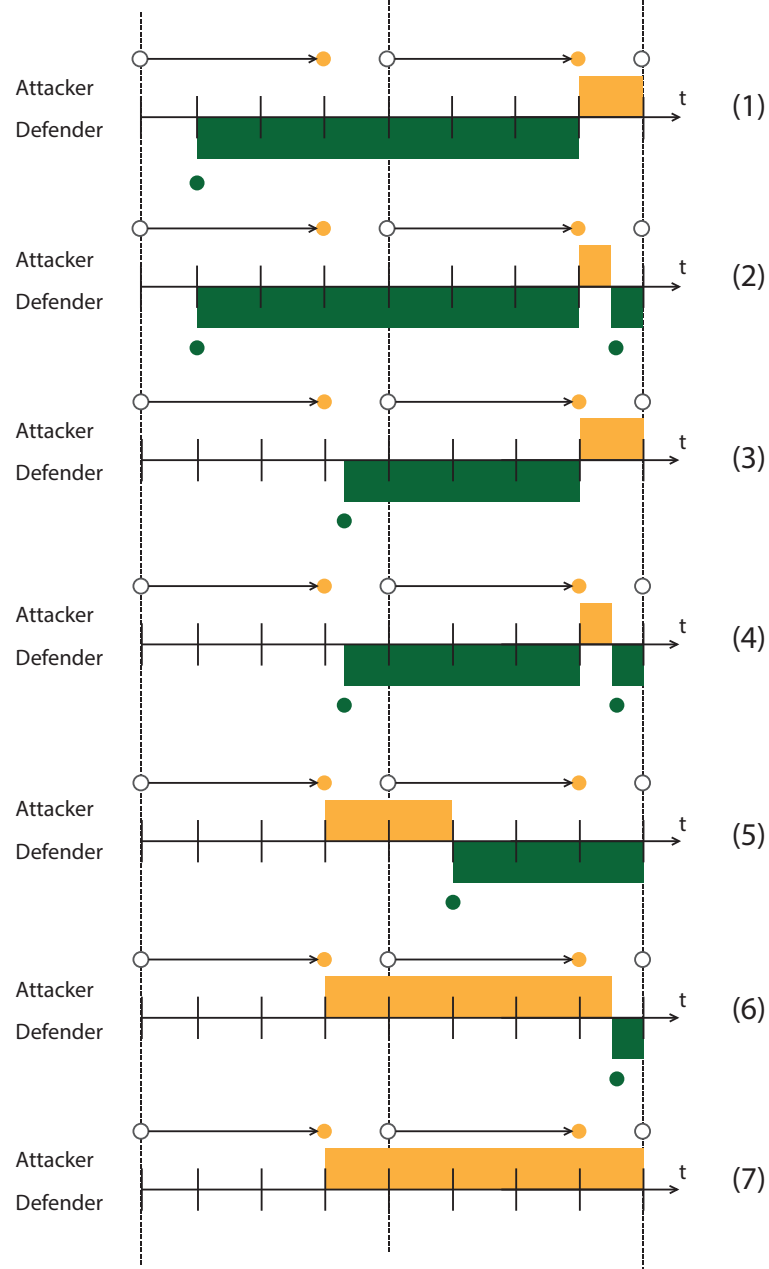


Figure 3.7: All possible cases for the attacker and the defender in Case 2.A where $d + \delta_A < \delta_D$. As can be seen in cases (1) to (4), the defender will have control during a period of d over the resource in the next interval when the defender has flipped in the previous interval.

probability is 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. 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:

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

This yields in the following benefit formula:

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

The benefit for the attacker will be as follows:

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

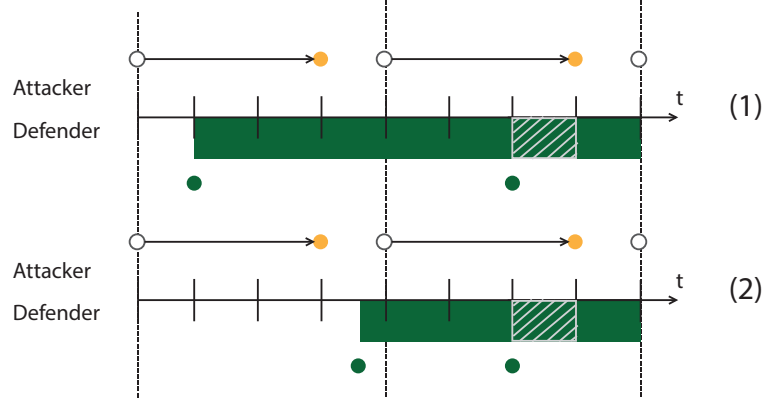
It is crucial that δ_D is at least as large as $d + \delta_A$. If not, this would mean that the defender can move during the delay in the interval following the interval where the defender already moved. This would mean that there can be an overlap between the average gain of $\frac{\delta_A}{2} + d$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

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

To obtain the formula in case of a too long delay, we therefore need to subtract this overlapping gain from the above formula. Since $\delta_D \geq \delta_A$, if the defender enters the interval immediately after the attacker has played, then the defender cannot have played in the previous interval. In that case, there is no overlap. So the problem of the overlap only appears if the defender enters late enough and thus only the last

3. FLIPIT WITH PROPAGATION DELAY

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



part of the delay is subject to overlap. The larger the difference between the interval of the defender and the attacker, the smaller the risk of overlap. Concretely, only the last part of length $d - (\delta_D - \delta_A)$ is subject to overlap. Hence, the probability of overlap is $\frac{d - (\delta_D - \delta_A)}{\delta_D}$ and the gain will be half of this interval: $\frac{d - (\delta_D - \delta_A)}{2}$. The gain rate to be subtracted is therefore:

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

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

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

This yields in the following benefit formula:

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

The benefit for the attacker will be as follows:

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

Hoofdstuk 4

Nash Equilibria

4.1 Nash Equilibria

As a second step, we are interested in finding Nash equilibria, points for which neither player will increase his benefit by changing his rate of play. More formally, a Nash equilibrium for the periodic game is a point (α_0^*, α_1^*) such that the defender's benefit $\beta_0(\alpha_0, \alpha_1^*)$ is maximized at $\alpha_0 = \alpha_0^*$ and the attacker's benefit $\beta_1(\alpha_0^*, \alpha_1)$ is maximized at $\alpha_1 = \alpha_1^*$. To begin with, some useful notation. We denote by $\text{opt}_0(\alpha_1)$ the set of values (rates of play α_0) that optimize the benefit of the defender for a fixed rate of play α_1 of the attacker. Similarly, we denote by $\text{opt}_1(\alpha_0)$ the set of values (rates of play α_1) that optimize the benefit of the attacker for a fixed rate of play α_0 of the defender. .

4.1.1 Determining the piecewise functions $\text{opt}_D(\delta_A)$

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 . To determine $\text{opt}_D(\alpha_A)$ we need to compute the derivative of $\beta_D(\alpha_D, \alpha_A)$ for a fixed α_A . We consider two cases:

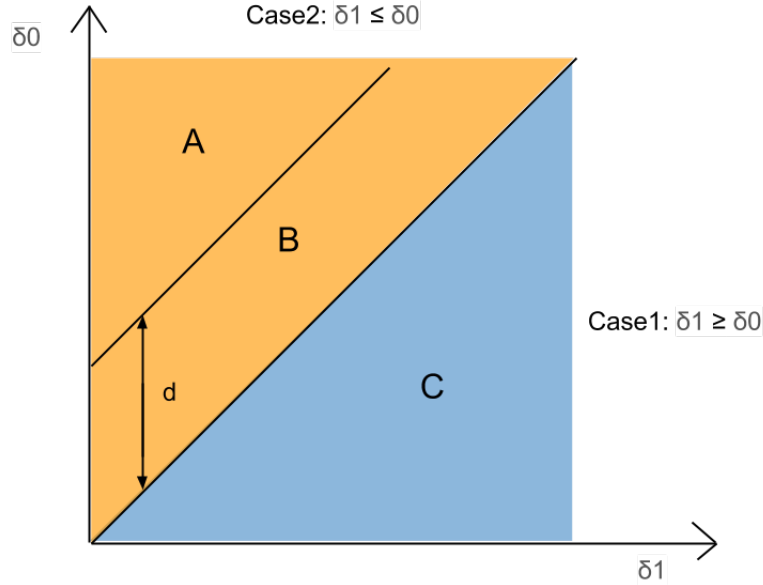
Case 1: $\delta_D \leq \delta_A$

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

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

If we take the partial derivative for a fixed δ_A we get the following result:

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



Figur 4.1: This figure shows the three subcategories of each case. 'A' stands for Case 2.A: $\delta_D \geq d + \delta_A \geq \delta_A$ and 'B' stands for Case 2.B: $d + \delta_A \geq \delta_D \geq \delta_A$

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

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

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 for this case is as follows:

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

Given that δ_D is always positive, the benefit function can be either positive or negative depending on the numerator of the above fraction. For $\delta_A + 2(d - k_D) > 0$

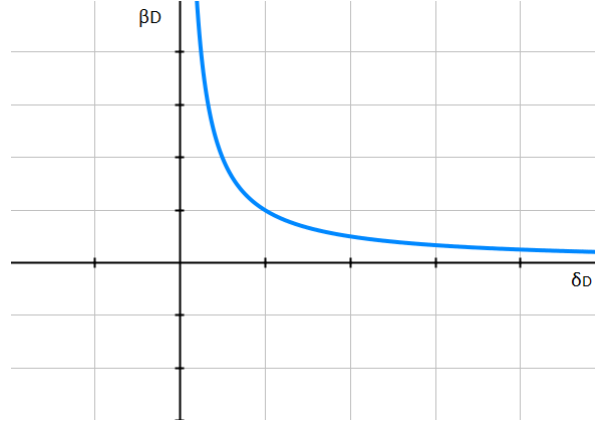


Figure 4.2: The benefit function is of the shape of $1/x$ if δ_D is always positive and if $\delta_A + 2(d - k_D) > 0$.

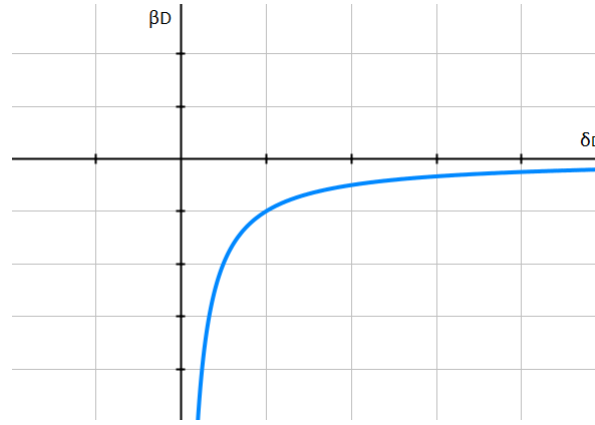


Figure 4.3: The benefit function is of the shape of $-1/x$ if δ_D is always positive and if $\delta_A + 2(d - k_D) < 0$.

the benefit will be always positive but decreasing, see figure 4.2.

The defender will play if $\delta_A > 2(k_D - d)$ and $k_D > d$. The defender will not play if $k_D < d$ because then δ_A is negative.

For $\delta_A + 2(d - k_D) < 0$, the benefit will always be negative so the defender will not play. See figure 4.3. This is independent of the value of k_D and d .

This leads to the following deduction:

If $k_D \leq d$

decreasing $\delta_A < 2(k_D - d)$, δ_A is negative

increasing $\delta_A > 2(k_D - d)$

If $k_D > d$

increasing $\delta_A < 2(k_D - d)$

decreasing $\delta_A > 2(k_D - d)$

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

The benefit formula obtained previously for this case is as follows:

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

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

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

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

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

This leads to the following deduction: The function increases on $[0, \sqrt{2\delta_A k_D + d^2}]$ and is decreasing on $[\sqrt{2\delta_A k_D + d^2}, \infty]$. So there is a maximum on $\delta_D = \text{minimum}\{\delta_A, \sqrt{2\delta_A k_D + d^2}\}$

Best responses

The optimum functions will be piecewise functions. There will be different optimum functions depending on k_D and d . We distinguish for these two cases, three sub cases for different values of δ_A .

$$k_D \leq d$$

Because $k_D \leq d$, the term $2(k_D - d)$ will always be negative. We point out that δ_A and δ_D are positive rates.

- if $\delta_A < 2(k_D - d)$
This means that δ_A has to be negative which is not possible. For case the defender will not play.
- $\delta_A = 2(k_D - d)$
 δ_A is negative or equal to 0 so the attacker will not play. For case 1 and case 2.b the defender will also not play.
- $\delta_A > 2(k_D - d)$
Case 2.a it is increasing for every value $\delta_A \in [0, \infty]$. For case 1 together with case 2.b the optimal benefit is achieved at rate $\delta_D = \sqrt{d^2 + 2\delta_A k_D}$.

$k_D > d$

Because $k_D > d$, the term $2(k_D - d)$ will always be positive. We point out that δ_A and δ_D are positive rates.

- if $\delta_A < 2(k_D - d)$
From case 2.a it follows that the benefit of the defender increases. From case 1 and case 2.b together the optimal benefit of the defender is achieved at rate $\delta_D = \sqrt{d^2 + 2\delta_A k_D}$.
- $\delta_A = 2(k_D - d)$
From case 2.a it follows that $\beta_D(\delta_D, \delta_A) = 0$, for all $\delta_A \in [0, 2(k_D - d)]$. From case 1 and case 2.b together the optimal benefit for the defender is achieved for all rates $\delta_D \in [0, \sqrt{d^2 + 2\delta_A k_D}]$.
- $\delta_A > 2(k_D - d)$
For case 2.a the benefit is decreasing. From case 1 and case 2.b the best strategy for the defender is not playing at all.

From this analyses we can compute $opt_D(\delta_A)$ for two different cases as:

dat laatste nog eens nakijken

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

For case :

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

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

We still consider the case where $d < \delta_D$.

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 2 cases:

Case 1: $\delta_A \geq \delta_D$

The benefit formula obtained previously for this case is as follows:

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

The derivative for a fixed δ_D is as follows:

$$\frac{\partial \beta_A(\delta_D, \delta_A)}{\partial \delta_A} = -\frac{\delta_D}{2\delta_A^2} + \frac{k_A}{\delta_A^2} - \frac{d^2}{2\delta_D\delta_A^3} + \frac{d}{\delta_A^2}$$

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

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

It follows that $\beta_A(\delta_D, \cdot)$ is increasing if $2k_A < (\delta_D - d)^2/\delta_D$ and decreasing if $2k_A > (\delta_D - d)^2/\delta_D$.

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

The benefit formula obtained previously 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:

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

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 $\delta_A = \max\{\delta_D, \sqrt{2k_A\delta_D}\}$. The maximum between δ_D and $\sqrt{2k_A\delta_D}$ is needed because δ_A cannot exceed δ_D in this case.

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

The benefit formula obtained previously 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:

$$\frac{\partial \beta_A(\alpha_D, \alpha_A)}{\partial \delta_D} = 0 \quad \Rightarrow \quad 2k_A = \frac{(\delta_D - d)^2}{\delta_D}$$

it follows that $\beta_A(\delta_D, \cdot)$ is increasing if $2k_A < (\delta_D - d)^2/\delta_D$ and decreasing if $2k_A > (\delta_D - d)^2/\delta_D$. 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 and k_A .

For this term $\frac{(\delta_D - d)^2}{\delta_D}$, d has to be bigger than δ_D because the cost k_A 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 .

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

Then for case 1 and case 2.b the benefit of the defender is increasing. From case 2.a follows that the optimal benefit for the attacker is achieved at the rate $\delta_A = \delta_D$

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

From case 1 and case 2.b it follows that $\beta_D(\delta_D, \delta_A) = 0$, for all $\delta_A \in [0, \frac{(\delta_D - d)^2}{2k_A}]$. From case 2.a the optimal benefit for the defender is achieved for all rates $\delta_A \in [0, \delta_D]$.

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

All decreasing.

This brings us to the following opt function for : From this analyses we can compute $opt_D(\delta_A)$ for two different cases as:

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

juiste resultaten
hier nog invul-
len

kA met kD ver-
gelijken en nash
evenwichten
vinden

een voorbeeld
invullen met
specifieke waar-
den

Hoofdstuk 5

Models for the Delay

The formalisation of the FlipIt game with 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 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 where 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 is depending 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 the presentation to propagation models of worms.

The overview of this chapter will be as follow: 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 that presents models to determine the time a worm needs to infect a sufficient number of nodes. These results can be used for the completion of parameter d . Finally, in section 5.3, we introduce an easy method to calculate the delay of the propagation of a worm.

5.1 Methods of propagation

In the context of malware propagation, there are two kinds of APT's. First, there are APT's that launch an attack on just one target node of a network. The mechanism these APT's 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 APT's target multiple victims. These APT's 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. So 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, we will limit the presentation to worm propagation models for APT's.

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 for pick up by a random person), email, or malicious software through fishing or trojan horses. More important for determining our delay is the propagation strategy. Propagation is performed by determining the next nodes to spread the worm to. The following are common propagation methods:

Selective Random scanning: Some worms propagate by using the method of selective random targeting IP addresses. The worm selects randomly a part of the selected 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' and 'Slammer'.

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' and 'Nimda' worm.

Topological scanning: Address information stored in the victim machines are used to locate new targets. This method used by the 'Morris' worm.

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'.

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. There are some difficulties with this scanning technique. First it is difficult to obtain the whole address space from the DNS records. Secondly, the worm has to carry the database, which can slow down the propagation if

the database is too big. Last, the IP addresses stored in the address table are only hosts with public domain names.

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 ‘Spybot’ or ‘network Bluepill’.

Topology-based Worms Email and other client application worms: An email worm 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’ is a Javascript computer worm that spread itself by exploiting a bug in Outlook Express.

5.2 Models for worm propagation

To use the models for worm propagation in our model of the FlipIt game it is sufficient to know the propagation speed of the worm and define after how many time units the network is defected. This will be equal to the d parameter in our model.

The methods of propagation can be divided into different model types. We are only interested in models that are based on SI model type (Susceptible-Infected), because in our model every node that is infected will stay infected. In table 5.1 the taxonomy of the propagation methods are given and the type of model.

For more information about other models the reader is referred to the following papers with other most common propagation models: [21], [15], [23] and [17].

5.2.1 Continuous-time models

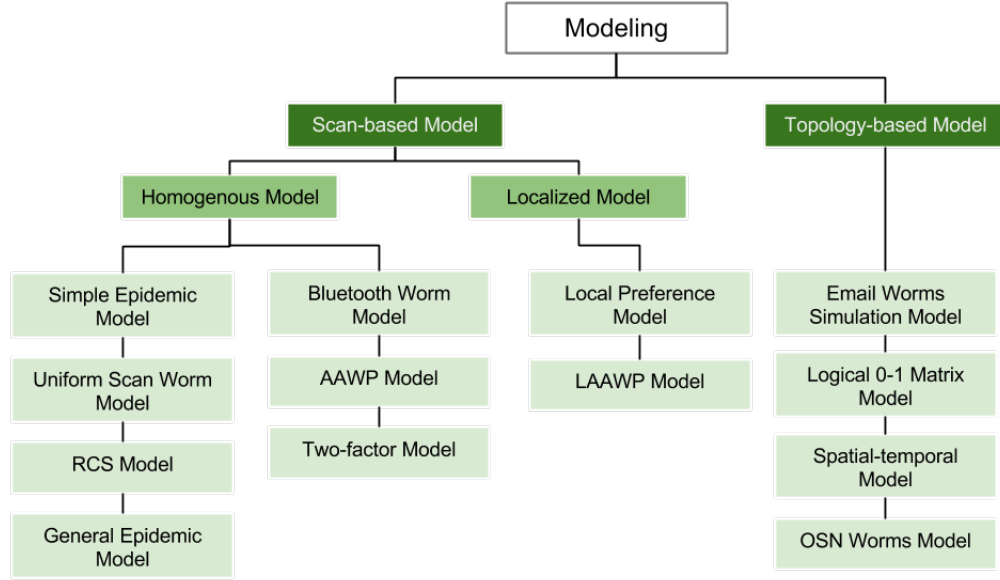
Simple Epidemic model

A Simple epidemic model is SI model. Nodes in the network can be either susceptible or infected. Once a node is infected it will stay infected. Every node has the same chance to be infected. The simple epidemic model is considered to be of a fixed size. The model for a fixed population is as follows:

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

where $I(t)$ is the number of infected hosts at time t ; β is the propagation rate; and N is the number of nodes in the network. In the beginning, $t = 0$, $I(0)$ hosts are

5. MODELS FOR THE DELAY



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;

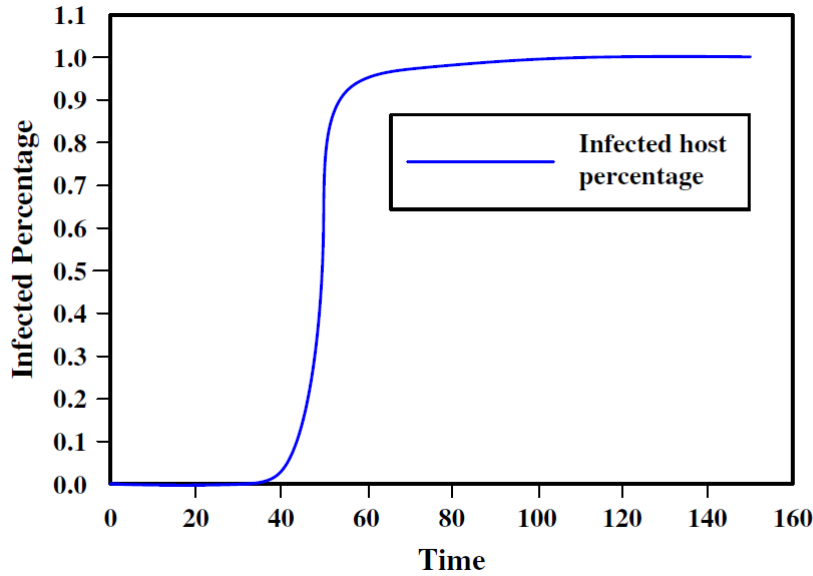
SI: susceptible-infected model; SIR: susceptible-infected-recovered model; SIS: susceptible-infected-susceptible model

Figur 5.1: Taxonomy of worm modelling. Image based on taxonomy given in [21]

infected. All the other nodes, $N - I(0)$, in the network are susceptible. The solution of this equation is the following logistic curve:

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

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



Figuur 5.2: Simple Epidemic Model. The x-coordinate is the propagation time and the y-coordinate is the infected percentage of the whole network. Original image from [15]

Figuur 5.2 laat het SEM model zien. This Simple epidemic model has been used in various papers [19], [24] to model random scanning worm such as Code Red and Slammer.

For our paper we have to approximate the propagation speed en Infect gelijkstellen aan het aantal genoeg nodes die geïnfecteerd moeten zijn door de APT om de controle te hebben. Dan weten we hoelang het duurt. In het begin zullen er ook geen geïnfecteerde hosts zijn, dus de vergelijking kan opgelost worden.

Er wordt geen rekening gehouden met de topology ? nog uitzoeken.

Kermack-Mckendrick model: SIR

In the Kermack-Mckendrick model, also known as SIR model, nodes can have three states: susceptible, infected or removed. Once a node of the network has been recovered from a worm, the node will stay in the removed mode and never becomes infected again. These nodes are not able to infect other nodes and can no longer be infected.

Let $I(t)$ be the number of infectious hosts at time t , $R(t)$ be the number of removed hosts at time t and $J(t)$ is the number of infected hosts by time t , regardless the fact that a node can be in a removed state.

$$J(t) = I(t) + R(t). \quad (5.3)$$

The Kermack-McKendrick model can be represented as follows:

$$\left\{ \begin{array}{l} \frac{dJ(t)}{dt} = \beta J(t)[N - J(t)] \\ \frac{dR(t)}{dt} = \gamma I(t) \\ J(t) = I(t) + R(t) = N - S(t) \end{array} \right\} \quad (5.4)$$

Parameter β is again the rate of infection and γ is the rate of removal of infected hosts. $S(t)$ is the number of susceptible hosts at time t .

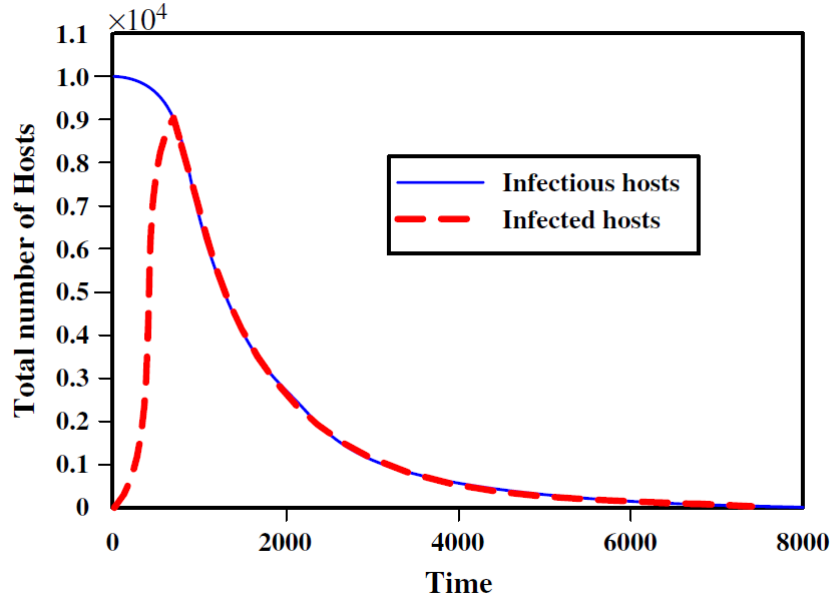


Figure 5.3: KM Model. The x-coordinate is the propagation time and the y-coordinate is the infected percentage of the whole network. Original image from [15]. $N = 10000, \beta = 1/10000000$.

—Define $\rho = \gamma/\beta$ to be the relative removal rate [3]. One interesting result coming out of this model is $dI(t)/dt > 0$ if and only if $S(t) > \rho$. (6) Since there is no new susceptible host to be generated, the number of susceptible hosts $S(t)$ is a monotonically decreasing function of time t . If $S(t_0) < \rho$, then $S(t) < \rho$ and $dI(t)/dt < 0$ for all future time $t > t_0$. In other words, if the initial number of susceptible hosts is smaller than some critical value, $S(0) < \rho$, there will be no epidemic and outbreak [15]. The Kermack-McKendrick model improves the classical simple epidemic model by considering that some infectious hosts either recover or die after some time.

However, this model is still not suitable for modeling Internet worm propagation. First, in the Internet, cleaning, patching, and filtering countermeasures against worms will remove both susceptible hosts and infectious hosts from circulation, but KermackMckendrick model only accounts for the removal of infectious hosts. Second, this model assumes the infection rate to be constant, which isn't true for a rampantly spreading Internet worm such as the Code Red worm—

self disciplinary worms

Model for self disciplinary worms and counter measures ... []

5.2.2 Continious-time models

– notities hier op overschrijven –

OSN worms model

Email worms simulation model

5.3 methode met matrixen

In this chapter we propose a method to calculate the spread of the worm in 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. The method is based on the method for Google Page Ranking [3]. For some of the spreading 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.

The computer network can be modelled by an undirected Graph $G = \langle V, E \rangle$ where $|V|$ denotes the number of resources in the network and $|E|$ the number of connections. We can convert this to an adjacency matrix which represents which vertices of the graph are neighbours of other vertices.

The graph is represented as a $|V| \times |V|$ matrix with for every entry a_{ij} a 1 as value if there is a connection between node V_i and V_j and 0 otherwise, and with 0's for every a_{ii} . Because the graph is undirected we have a symmetric matrix.

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: [22] .

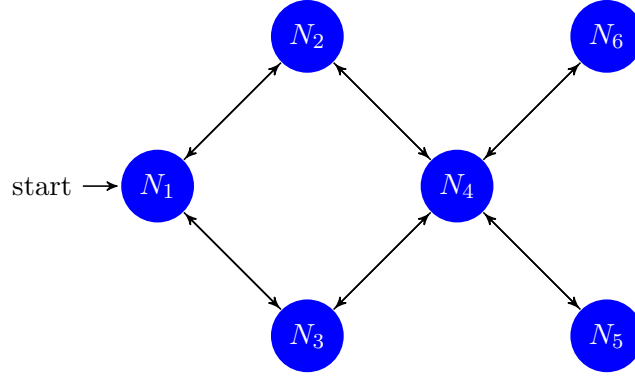


Figure 5.4: Network with 6 nodes. The arrows represent the connections between the nodes. The start point is where the worm has been dropped, here node 1.

Assuming a network like in figure 5.4, the corresponding adjacency matrix is the matrix $[A]$:

$$\begin{array}{c} N_1 \quad N_2 \quad N_3 \quad N_4 \quad N_5 \quad N_6 \\ \begin{array}{l} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{array} \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

In matrix $A \times A = A^2$, each entry represents the number of 2 step paths from N_i to N_j : We denote this matrix as matrix $[B]$

$$\begin{array}{c} N_1 \quad N_2 \quad N_3 \quad N_4 \quad N_5 \quad N_6 \\ \begin{array}{l} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{array} \begin{pmatrix} 2 & 0 & 0 & 2 & 0 & 0 \\ 0 & 2 & 2 & 0 & 1 & 1 \\ 0 & 2 & 2 & 0 & 1 & 1 \\ 2 & 0 & 0 & 4 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

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 : We denote this matrix as matrix $[C]$

$$\begin{array}{c} N_1 \quad N_2 \quad N_3 \quad N_4 \quad N_5 \quad N_6 \\ \begin{array}{l} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{array} \begin{pmatrix} 0 & 4 & 4 & 0 & 2 & 2 \\ 4 & 0 & 0 & 6 & 0 & 0 \\ 4 & 0 & 0 & 6 & 0 & 0 \\ 0 & 6 & 6 & 0 & 4 & 4 \\ 2 & 0 & 0 & 4 & 0 & 0 \\ 2 & 0 & 0 & 4 & 0 & 0 \end{pmatrix} \end{array}$$

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

Using this knowledge we can calculate in how many steps a node is infected: The infection vector or start vector I_0 of length $|V|$ indicates which node is infected and which not. Multiplying the infection vector with A results in a vector I_1 , which represents which nodes are infected after 1 step. Multiplying the start vector I_0 with A^N calculates which nodes are infected in N steps: each non zero entry represents an node infected after N steps.

In the context of the FlipIt game, it is safe to assume that once a node is infected, it stays infected until the defender Flips the node. This means that after d steps, it is assumed that all nodes that could be reached in less than d steps from the node where the worm was dropped, are infected. In order to know how many nodes are infected after (for example) at most 3 steps, we have to consider nodes that are infected initially (step 1), or after 2 steps, or after 3 steps. Calculating the sum of the three matrices $(A + A^2 + A^3)$ results in a matrix that indicates for each node, the number of paths of length 1, 2 or 3 from i to j . Multiplying the start vector with this matrix, results in a vector that indicates which nodes will be infected in at most 3 steps. This technique can be applied to calculate the state of the network after any number of steps.

Determining the length of the delay boils down to determining which configuration of infected nodes is considered as corresponding to the attacker having flipped the resource. It may be that all nodes need to be flipped, a sufficient amount of nodes, or a (set of) particular nodes.

–Hier nog een beetje verder uitschrijven hoe je exact de d dan moet bepalen, gegeven dat je niet weet in welke knoop het virus gedropt wordt–

Hoofdstuk 6

Conclusion

This paper presents an adaption to the basic FlipIt game by [20] to model an attacker with a delay. We have

The work presented in this paper represents an initial calculation 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.

..

6.1 trala

Bibliografie

- [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] Cornell. Google page rank, 2009.
- [4] X. Feng, Z. Zheng, P. Hu, D. Cansever, and P. Mohapatra. Stealthy attacks meets insider threats: A three-player game model.
- [5] Kaspersky. A business approach to managing data security threats. In *IT security risk survey 2014*, 2014.
- [6] Kaspersky. Special report on mitigation strategies for advanced threats. In *The power of protection*, 2014.
- [7] A. Laszka. Flipthem: Modeling targeted attacks with flipit for multiple resources. *5th International Conference, GameSec 2014, Los Angeles, CA, USA, November 6-7, 2014. Proceedings*, 8840:175–194, 2014.
- [8] A. Laszka, B. Johnson, and J. Grossklags. Mitigating covert compromises. *iets*, 8289:319–332, 2013.
- [9] A. Laszka, B. Johnson, and J. Grossklags. Mitigation of targeted and non-targeted covert attacks as a timing game. 8252:175–191, 2013.
- [10] K. Leyton-Brown and Y. Shoham. *Essentials of Game Theory: A Concise, Multidisciplinary Introduction*. Synthesis lectures on artificial intelligence and machine learning. Morgan & Claypool Publishers, 2008.
- [11] Y. S. M. Jackson, K. Brown. Coursera game theory. Stanford University and The university of British Columbia, 2004.

- [12] Microsoft. security bulletin release.
- [13] 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.
- [14] 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.
- [15] S. Qing and W. Wen. A survey and trends on internet worms. *Computers & Security*, 24(4):334–346, 2005.
- [16] 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.
- [17] G. Serazzi and S. Zanero. Computer virus propagation models. In *Performance Tools and Applications to Networked Systems*, pages 26–50. Springer, 2004.
- [18] W. Stallings. *Network security essentials: applications and standards*. Pearson Education India, 2007.
- [19] 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.
- [20] M. van Dijk, A. Juels, A. Oprea, and R. Rivest. Flipit: The game of "stealthy takeover". *Journal of Cryptology*, 26(4):655–713, 2013.
- [21] 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.
- [22] Wikipedia. Adjacency matrix, 2015.
- [23] 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.
- [24] C. C. Zou, W. Gong, and D. Towsley. Code red worm propagation modeling and analysis. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 138–147. ACM, 2002.

Fiche masterproef

Student: Sophie Marien

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

Engelse 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 telcom), 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 there 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