TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Science
TUT Centre for Digital Forensics and Cyber Security

ITC70LT
Kristo Kapten 132719IVCMM

# THREAT MODELING FRAMEWORK FOR HOME GAMING CONSOLES

Master's thesis

Supervisor: Hayretdin Bahsi
PhD.
Senior Research Scientist

Tallinn 2016

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutiteaduste instituut
TTÜ Digitaalse ekspertiisi ja Küberkaitse keskus

ITC70LT
Kristo Kapten 132719IVCMM

# OHU MODELLEERIMISE RAAMISTIK KODU MÄNGUKONSOOLIDELE

Magistritöö

Juhendaja: Hayretdin Bahsi
PhD.
Vanemteadur

Tallinn 2016

# Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Kristo Kapten

25.05.2016

# Abstract

The main purpose of this thesis is to create a proof-of-concept Threat Modeling framework called Home Gaming Consoles Threat Modeling (HGCTM) for determining potential threats and deducting test cases for back box testing, as their design and requirements specific documentation is unavailable to the public. The methodology behind the framework bases on the analysis of existing threat modeling solutions and theoretical research on threat modeling principles. It consists of multiple phases, such as modeling the system, determining potential threats, addressing the threats and creating test cases. The analysis of the created framework in terms of applicability and usability is done on the most popular eighth generation home gaming console - the PlayStation 4. The initial results of the PlayStation 4 network features test case, following the HGCTM framework, prove the viability of the concept. Therefore, the goals of this thesis were ultimately met. The resulting framework has many possibilities for future improvements. The most notable developments will be made on the Threat Modeling Tool's HGCTM template and the creation of test cases.

The thesis is in English and contains 58 pages of text, 7 chapters, 8 figures, 5 tables.

# Annotatsioon

## Ohu Modelleerimise raamistik Kodu Mängukonsoolidele

Käesoleva lõputöö põhieesmärgiks on luua Ohu Modelleerimise raamistik Kodu Mängukonsoolide Ohu Modelleerimise (HGCTM) kontseptsiooni tõestamise versioon, et tuvastada potentsiaalseid ohte ja tuletada musta kasti testjuhtumeid, sest nende disaini ja tehniliste nõuete dokumentatsioon pole avalikkusele kättesaadav. Kontseptsioon keskendub peamiselt kaheksanda põlvkonna konsoolidele ja nende võrgupõhistele omadustele. Raamistik kasutab ründavat perspektiivi, et teha süsteemi funktsioonidest struktureeritud ja korrataval viisil ohu modellerimist. HGCTM raamistiku rakendamine on piisavalt lihtne igale küberkaitse testijale või teadlasele.

Raamistiku tagune metoodika põhineb olemasolevate ohtu modelleerimise lahenduste ja ohu modelleerimise põhimõtete teoreetilise uuringu analüüsil. See koosneb mitmetest faasidest, nagu süsteemi modelleerimine, potentsiaalsete ohtude tuvastamine, ohtude käsitlemine ning test juhtumite loomine.

Loodud raamistiku rakendatavuse ja kasutatavuse analüüs tehti kõige populaarsema kaheksanda põlvkonna kodu mängukonsoolil - PlayStation 4'l. Esialgsed PlayStation 4 võrguomaduste testitulemused, järgides HGCTMi raamistikku, tõestavad kontseptsiooni elujõulisust. Seega, käesoleva lõputöö eesmärgid said lõplikult täidetud.

Loodud raamistikul on mitmeid võimalusi tulevasteks parandusteks. Kõige märkimisväärsemaid edasiarendusi tehakse Ohu Modelleerimise Tööriista HGCTMi mallil ja test juhtumite loomisel. Esiteks, mall hakkab sisaldama juba seadistatud šabloone mis on seotud kaheksanda põlvkonna konsoolidega. Teiseks, Kuritarvitamise juhtumite loomine hakkab olema palju sujuvam Ohu Modelleerimise Tööriista tulemuste poolest.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 58 leheküljel, 7 peatükki, 8 joonist, 5 tabelit.

# List of abbreviations and terms

| | |
|---|---|
| ASF | Application Security Frame |
| CPU | Central Processing Unit |
| DFD | Data Flow Diagram |
| DREAD | Threat-risk ranking model (Damage Potential, Reproducibility, Exploitability, Affected Users, Discoverability). |
| GPU | Graphics Processing Unit |
| HGCTM | Home Gaming Consoles Threat Modeling |
| LAN | Local Area Network |
| Microsoft SDL | Microsoft Security Development Lifecycle |
| OWASP | Open Web Application Security Project, a non-profit security-oriented online community. |
| PC | Personal Computer |
| PS4 | PlayStation 4, home gaming console developed by Sony Computer Entertainment (SCE) and is the successor to the PlayStation 3. |
| STRIDE | Threat categorization and determination method according to attacker's goals (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege). |
| TMT | Microsoft's Threat Modeling Tool |
| Wii U | Home gaming console developed by Nintendo and is the successor to Wii. |
| XB1 | Xbox One, home gaming console developed by Microsoft and is the successor to the Xbox 360. |

# Table of contents

# List of figures

# List of tables

# 1 Introduction

Today's home video gaming consoles are more than just gaming consoles, but rather the whole entertainment and multimedia centers within the living rooms. With all the added features and possibilities in each new generation of home gaming consoles, they resemble more with personal computers (PCs) than ever before. With the latest eighth generation, they are basically computers by another name, but unfortunately with much of the same cybersecurity concerns.

Although Nintendo's Wii U video game console was released almost a year before Sony's PlayStation 4 (PS4) and Microsoft's Xbox One, the two devices have gained more attention regarding their network services and good hardware standards than the Wii U. This is because both, the PS4 and the Xbox One, utilize x86 processors that drastically boost performance compared to their previous generations and what are known to be used in PCs and Macs. Unfortunately, they may also introduce new attack surfaces for hackers skilled in such architectural exploits [1]. Even more disturbing is the fact that they may have the ability to use already existing tools and exploits created for x86 systems.

The concerns are already met, as there have been developments in regards to breaking these systems security. In December 2015, it was showcased at the CCC hacking convention, that the hacking crew known as the Fail0verflow, manages to run a Linux operating system on the PlayStation 4. With this achievement however, it was stated that they will not share any hacks or exploits they used to install or make it run [2]. Fortunately, this at the time meant that, unless you weren't a hacker with your own PS4 Kernel exploit, this information wouldn't be much of use [3].

Several months later however, on 2nd March 2016, the first publically available PS4's kernel exploit named as BadIRET was leaked [4]. As of that moment, it opened up the possibility for homebrew systems and potential piracy on the PlayStation 4 to a wider audience. What's even worse, the exploit was built upon the vulnerability found in WebKit, an open source web browser engine used in the PS4 [5]. Even though, the vulnerability seems to be fixed after the PS4 firmware version 1.76 – it shouldn't be taken

lightly, because a direct connection between the system exploitation, vulnerability and a network feature like the web browser and its engine was made.

When looking into Xbox One's (XB1) security incidents, it seems that Microsoft has done a far better job in building a secure gaming console than its competitors. This is mainly due the lack of news regarding Xbox One's security or hacking. In overall, there are some rumors around possible hacks being developed and games pirated, but nothing is fully confirmed. From the hacking perspective, it said that an exploit for XB1 may be in the works by the Team Xecuter, but nothing has been released [6]. On the topic of piracy, there may be a way to copy already installed games and licenses onto another Xbox's hard drive, but due to the news originating from Indonesia, there's lack of evidence [7].

Wii U, the competitor of XB1 and PS4 may be less popular in overall, but similar security concerns are also present in Nintendo's systems. It is known that only months after the release, in 2013, hackers claimed to have cracked the system's security regarding encryption keys and disk authentication, which are used to ensure that only trusted sources and games are used [1]. As a result, this breach could allow users to pirate games via USB drive by bypassing security and validation checks that come with the trusted sources like optical discs or purchases from the Nintendo's eShop. It may seem harmless and cost-saving for some users, but in reality, doing so puts their gaming console and personal information at risk when removing these safety measures which protect them against malicious software.

Now, when looking at the current evidence of the eighth generation gaming consoles security issues, it can be reasoned that even the latest and greatest home video gaming consoles are vulnerable to hackers and cybercriminals. Moreover, as all three, the PS4, XB1 and Wii U, use more online services than their respective predecessors [1], it opens up even more pressing security concerns that need to be addressed before they are realized. As such, security researchers and testers should analyze and test these systems against potential threats to discover possible vulnerabilities before cybercriminals use them in the wild.

## 1.1 Motivation

Currently almost 70 million eighth generation home video gaming consoles have been sold since 2012, but even more gaming consoles are being used thanks to the previous generations [8], [9], [10]. As home gaming consoles aren't just for gaming anymore, different features like built-in web browsing, media streaming, wireless communications and peripherals can open up potential vulnerabilities in these systems.

To make these latest generation gaming consoles even more vulnerable, it is almost a necessity for them to be connected to the Internet, as majority of features require network connection to be usable. This in turn opens up a huge possibility for hackers with malicious intents to try and exploit vulnerabilities within these network connected systems. Even when only one good vulnerability is discovered, then millions of gaming consoles could be at risk. That is why, it is paramount to research and test whether they are vulnerable to any network based threats.

Security researchers, testers and others who have looked into gaming consoles security, are using different approaches and methodologies to identify, address or test the systems against potential threats. This is done either by best to their knowledge, following analogous devices vulnerabilities or relying completely on scanning and testing tools that weren't designed for gaming consoles. Thus, leaving the possibility for each test case to miss some potential threat areas or focus too much energy on irrelevant or already mitigated areas.

For example, a researcher Walter W. Ridgewell explains in his Master's paper "Determination and Exploitation of Potential Security Vulnerabilities in Networked Game Devices" that, when looking for unknown vulnerabilities in networked game devices and not having previous game device specific framework to go by – he is using the vulnerabilities of personal computers, mobile phones and their respective applications, as a framework for his research [11, p. 27]. This gave him a way to start the research with known security vulnerabilities, but it can be quite certain that few of the threat areas were left unexplored. Moreover, he should have used an approach that not only covers all threat areas, but provides exact test cases for testing and easy repeatability for future improvement in the research.

Another Master's thesis "Game Consoles - Are they secure?" by Halvar Myrmo relies its research and technical experiments solely on open source software like Nmap, Nessus, Wireshark, Metasploit and SPSS, because the author was not able to find any previous work in this area that had conducted similar experiments [12, pp. 15-24]. This was also the reason why it was decided not to go in very detailed technical experiments on one platform, but decided to do a wider survey of different gaming consoles. This approach clearly states that, when there's no framework to follow, next best is quantitative analysis by available software. With the quantitative analysis however, there are bound to be areas that are left untouched and the quality of the result could be questionable.

Even though, the reviewed research papers are about previous generations of gaming consoles, both authors could have benefited greatly when they had used similar approach to continue and improve the research of others. At the time, they couldn't find a good solution, but the issue seems to be still relevant today. This is because the author of this thesis wasn't also able to find a suitable framework or methodology that could easily help determine and create black box test cases for potential threats in regards to gaming consoles.

## 1.2 Contribution

For cybersecurity researchers or testers to test systems against possible vulnerabilities, there's usually a methodology or a way on how to determine potential threats regards to the specific system. Not only determine, but to rate, prioritized and create test cases for most relevant threats. Moreover, there is a need for a framework so that all the threat areas are covered in a structured way. This also provides repeatability and possibility for future improvements or collaboration on the same models. As the author couldn't find any good unified methodology or approach that suits the needs described and can be used on home gaming consoles or similar systems, one should be created.

Therefore, there is a need for a solution that:

- helps determine potential threats in regards to home gaming consoles;
- covers all threat areas in a structured way;
- rates and prioritizes found threats;
- creates visual models for test cases;

- is repeatable for further improvements or collaboration;
- easy enough for researchers or testers to implement.

This given solution helps researchers and testers to cover all threat areas, focus only on potential threats and use created visual models for black box testing. Even though, the implementation of this solution takes time, the work done in this phase saves a lot of time in comparison to testing all the possible threat scenarios. This solution can be used or further improved by individuals or by communities thanks to the standardized and repeatable approach.

## 1.3 Scope

The goal of this thesis is to create a proof-of-concept threat modeling framework, that helps determine potential threats and create test cases for back box testing in regards to home gaming consoles. The conceptual framework is called Home Gaming Consoles Threat Modeling (HGCTM). The scope of this concept focuses on eighth generation gaming consoles and their network based features. As such, only software based threats are considered. Moreover, the framework uses adversarial perspective to perform threat modeling on the system's features in a structured and repeatable manner. As the framework itself is limited to proving the overall concept, the samples within the implementation phase are simplified for better comprehension.

## 1.4 Outline

In general, this thesis can be divided into five main sections:

1. Overview on eighth generation gaming consoles and their preliminary analysis;
2. Analysis of different threat modeling approaches;
3. Threat modeling principles and creation of the Home Gaming Consoles Threat Modeling framework;
4. Implementation of the proposed framework;
5. Analyzation of the framework.

Firstly, the background chapter of the thesis gives an overview of the eighth generation gaming consoles and their network based features. To understand more, the preliminary

forensic analysis of the two most popular eighth generation gaming consoles provide the insight into the systems and their security measures.

Secondly, the analysis of the different threat modeling approaches gives an insight into the best known methodologies and their key differences. Additionally, the comparative analysis of these approaches concludes the need for a new way of achieving the thesis goals.

Thirdly, the principles of threat modeling and knowledge from previous section provides the conceptual basis for Home Gaming Consoles Threat Modeling framework's creation. Also, the chosen methodologies and tools for each principle are described according to thesis goals and needs.

Fourthly, the implementation phase of the framework shows its practical applicability and discusses each step in terms of chosen methodologies and tools needed for achieving the goals.

Finally, the analyzation of the framework discusses the applicability and usability of the proposed framework. In addition, possible limitations and issues are examined and solutions suggested. In the end, a comparative analysis between experimental approaches and the framework proves its superiority.

The conclusion summarizes the most important facts of the thesis and shares future improvements to the concept.

# 2 Background

In this chapter an overview of the eighth generation gaming consoles and their well-known network based features is given. Moreover, the preliminary forensic analysis survey of the two most beloved gaming consoles provides insights into their systems design and security.

## 2.1 Eighth generation gaming consoles

Home gaming consoles eighth generation began on November 18, 2012 with the release of Nintendo's Wii U and followed almost a year later with the releases on November 15, 2013 for Sony Computer Entertainment's PlayStation 4 and Microsoft's Xbox One on November 22, 2013 [13], [14], [15]. All three gaming consoles are considered as rivals, but due to the big differences in hardware, software and sales numbers, they can be easily distinguished. A quick overview of these gaming consoles starting from the most popular to the least, gives a glimpse of the systems and their attractiveness to the attackers.

### 2.1.1 PlayStation 4

The PlayStation 4 (PS4) is the most popular eighth generation home gaming console with over 35,9 million units sold worldwide as of January 3, 2016 since its release in November 2013 [8]. The gaming console is developed by Sony Computer Entertainment (SCE) and is the successor to the PlayStation 3 (PS3), but in contrary to the predecessor, the PlayStation 4's hardware is built around low power x86-64 personal computer (PC) like architecture. As such, the processor is developed by Advanced Micro Devices (AMD) in coordination with Sony and the central processing unit (CPU), graphics processing unit (GPU), as well as the memory controller and video decoder are all incorporated into single-chip custom processor [16].

The hardware in overall is designed to handle high performance tasks as well as to give the console longevity with the CPU consisting of "Jaguar" modules totaling in 8 cores. Additionally, the GPU consists of 1152 cores and is capable of producing a theoretical peak performance of 1.84 TFLOPS. Moreover, the system also contains 8 GB of GDDR5 memory which is capable for maximum clock frequency of 5500MHz and has a maximum memory bandwidth of 176 GB/s [17]. Similarly, to personal computers, it has

a built-in hard disk drive, optical drive capable of reading Blu-ray and DVD drives, input-output of Super-Speed USB, AUX, HDMI and for network communication it uses Ethernet, Wi-Fi IEEE 802.11 b/g/n and Bluetooth 2.1 (EDR) [18].

PlayStation 4 has many network based features, but two of the most used, which gives users the ability to interact with the gaming console remotely, are called Remote Play and Second Screen. The Remote Play allows users to play their PS4 games with their companion devices within the home local area network (LAN) or outside of it – over the Internet. The Second Screen feature is a bit more limited as it only displays unique content to the companion device while the games are played on the gaming console within the LAN [19].

Fortunately, the remote connection settings within the PS4 are set off by default, but after a few steps, they can me turned on. Turning them on, also means turning on the gaming console's rest mode, which keeps the gaming console always on stand-by and continuously connected to the Internet, as there might be a need for remote connection [20]. Both features are awesome for end-users, but could be even more interesting to cybercriminals, as these features may open up a possibility for network based attacks.

### 2.1.2 Xbox One

The Xbox One (XB1) is the second popular eighth generation home gaming console with the estimated 19 to 20 million units sold worldwide as of February 1, 2016 since its release in November 2013 [9]. The gaming console is developed by Microsoft and is the successor to the Xbox 360 and main competitor to PlayStation 4. The latter is mainly to the many similarities in both consoles.

In analogy to the PS4, the Xbox One's hardware is built around the same low power x86-64 "Jaguar" architecture, having the processor also developed by AMD. Thus, for both gaming consoles, the CPU and GPU are incorporated into single-chip custom processor. They have the same CPU with 8 cores, but differences start with the GPU. Sony decided to go with the 18 Compute Unit (CU) configuration, totaling in 1152 cores, where as Microsoft went for a much smaller configuration with 768 cores and 12 CUs. As such, Xbox One's GPU is capable of producing a theoretical peak performance of 1.23 TFLOPS in contrary to PS4's 1.84 TFLOPS. Similarly, the system also contains 8 GB of

memory, but the slower DDR3 version with 2133MHz, which has a maximum memory bandwidth of 68,3 GB/s [17].

To compensate the performance gap between the both gaming consoles, Microsoft decided to clock the CPU up to 1.75Ghz from 1.6GHz, which is still the default for PS4, rise the Xbox One's 800MHz GPU to 853MHz, and mitigate the system memory differentiation with 32MB ESRAM cache [21]. Even after all these measures, the XB1 still falls a bit short from the PS4's performance, because of the hardware differentiation.

On the storage and connectivity side, Xbox One has few additional options compared to PS4, but both are still very similar to PCs. It comes with a built-in hard disk drive, optical Blu-ray and DVD drive, input-output of three Super-Speed USB ports, HDMI in and out, S/PDIF out, IR-out and the Kinect port. Also, the network communications are done over Wi-Fi IEEE 802.11 b/g/n and Ethernet, but lacks of Bluetooth compared to PS4 and Wii U [22]. For the end-users, the missing Bluetooth connectivity option is definitely an annoyance, but in terms of security, it completely eliminates one possible attack surface for cybercriminals.

Identically to PlayStation 4, the Xbox One has also specific network features in addition to regular applications. For example, the very similar Remote Play and Second Screen functions, but which are a bit more limited in XB1. The remote play also known as the game streaming option is limited to local area network and is available only on Windows 10 PC's via the Xbox app [23]. Whereas the PS4's Remote Play function is usable over internet and available to PCs, Macs and other compatible devices [19], [20]. The Second Screen application called the SmartGlass lets users control their Xbox One console via companion device, but only when its turned on and connected to the SmartGlass device [24]. Even with the slight differences in features between gaming consoles, they both pose a possible security risk and should be investigated by the researchers in more detail.

### 2.1.3 Wii U

The Nintendo Wii U was released in November 2012 and is the first eighth generation home video gaming console [13]. It is a successor to the Nintendo Wii and competes with Sony's PlayStation 4 and Microsoft's Xbox One gaming consoles. Even though, Wii U was released almost a year before its competitors, it is the least owned console out of the three, with only 12,8 million units sold worldwide as of March 31, 2016 [10].

As with all the eighth generation home gaming consoles, the Wii U also has a custom built multi-chip module (MCM) that combines the CPU, GPU and the memory chip in one. It was developed in cooperation with IBM and like with all the other consoles, the GPU chip with AMD [25]. The console contains 2 GB of DDR3 system memory and is said to be 20 times the amount found in its predecessor the Wii [26]. This however, still isn't really comparable to its current competitors, as Wii U's hardware in overall is four to five times weaker. From the networking side, Wii U uses the same Wi-Fi IEEE 802.11 b/g/n standard for network communications like the rivals, but it doesn't have the cable option due to the lack of Ethernet port. Fortunately for users, it features four USB 2.0 connectors that support Wii LAN Adapters [27].

Even though, the Wii U was the first eighth generation home gaming console on the market, it has the overall weakest hardware and sales numbers compared to its competitors. This in turn makes it the least desirable target for the attackers and cyber criminals. Additionally, it has a deficiency of network features and isn't really up to par with PlayStation 4 and Xbox One. This is also the reason why Wii U is only lightly discussed within this research paper and not analyzed deeper.

## 2.2 Preliminary forensic analysis

As majority of gaming platforms have their own proprietary and unique operating systems, it makes the data retrieval and information gathering a real challenge for researchers, testers and hackers. This is why the following overview of forensic analysis of PS4 and XB1 helps to give an insight into the closed gaming console systems and some of the discovered security measures.

### 2.2.1 Forensic analysis of the PlayStation 4

The first forensic analysis of Sony's PlayStation 4 focuses on overall investigation of the gaming console and on identifying information sources that provide potential value. It also gives forensic guidelines for acquiring information with both online and offline issues. Additionally, it presents an investigative methodology that can be of guidance to researchers or others working with similar systems [28].

The analysis starts by studying related literature and a practical investigation of the PlayStation 4. It's followed by identifying the areas by who, what, when and where

aspects, that the researcher wishes to explore. This basically means assessing the scope of the research and setting the goals for the research. Next the step is examining the features of the device and detailing the ones found to be of interest. Once all these aspects are identified, the best practice guide for digital evidence methodology is used [28]. Unfortunately, the forensic analysis of PS4 do not cover network based features that would give an insight into the security measures used in network communications, but it does give some valuable info about the overall system.

The analysis gives a general understanding about the PlayStation 4's system and some security components. It is found out that the quantity and quality of information retrievable depends ultimately on the firmware version installed on the console and with each new version the security is tightened. Also, as Sony has turned off the possibility for users to downgrade PS4's firmware, it is impossible to restore the gaming console to previous versions that would give better access [28]. This means that investigators, researchers and testers have to start examining and build their knowledge with each firmware version the moment these systems are released. This is where the proposed threat modeling framework is useful for security researchers, as they can follow the framework to model the system and determine potential threats according to each firmware version. Additionally, this can show exactly which potential threats were mitigated in the given firmware and which are still present.

### 2.2.2 Forensic analysis of the Xbox One

The preliminary forensic analysis of the Xbox One aims be the first forensic approach in examining the Xbox One, which is currently the most powerful Microsoft's home gaming console. The outcome of the initial analysis of the Xbox One provides a set of hard drive images and unique files, on which the researcher's community can further investigate. It also provides a knowledge of added security measures compared to its predecessor with the adaption of new file types and additional encryption to the data, which in overall makes it harder for information and valuable artifacts retrieval [29].

The analysis follows the guidelines to forensically examining system as suggested by National Institute of Standards and Technology (NIST) and the methodology on acquiring valuable digital information using open-source tools. Overall, exploratory methods such as file carving, keyword searches, network forensics and file system analysis are performed. Analysis of the known NTFS filesystem did allow for file timestamps to be

recovered, and some encrypted network traffic was related back to which game was played [29]. Fortunately, in contrast to PlayStation 4 forensic analysis, the network communications are here also moderately investigated.

From the forensic discoveries, different digital artifacts like times when the user initially set up the console, system was restored, shutdown or what games and applications had been downloaded and used were found. No credentials like passwords or user names were discovered during the analysis, but the location for the user data's local storage is hypothesized to be in the AppUserStorage file, which can be of interest to researchers and hackers. Moreover, it is found out that the files and game network traffic are encrypted and/or compressed nature, as each application tested has its own level of security [29].

From the researcher's or attacker's perspective, this is definitely an interesting information that needs further investigation, as the application that's traffic isn't encrypted, could fall under a sniffing attack and the exchanged information can be read by an unauthorized party. Additionally, it is discovered that the native web browser application, the Internet Explorer, allows gathering network traffic like the user is browsing the web on their computer [29]. This means that the user is relying on the security of the website the user is visiting. In result, when secure communication isn't used between the website and the gaming console, the user could fall for the same sniffing or data tampering attacks.

After discovering these facts, the security researcher can use this information to follow the proposed solution and do a structured threat modeling about the selected network based applications or features. The determined, prioritized and modeled potential threats can be then tested and validated. Additional analysis can be done with each system or application update, as there is now basis for comparison to which threats have mitigated or emerged.

# 3 Related works

Doing threat modeling on different systems and their software during their development phase isn't anything unusual, but doing it on already developed and closed systems like home gaming consoles is. This chapter gives a brief overview of the most widely used threat modeling approaches and their key components. In the end, a comparison between the different methods is provided with the concluding indication that there is a need for additional approach that meets the needs of this work.

## 3.1 Microsoft Security Development Lifecycle (SDL)

The Microsoft Security Development Lifecycle (SDL) is a company-wide policy for software security assurance process since 2004, which consists of five phases and plays a vital part in embedding security and privacy into Microsoft's created software [30]. The five phases are as follows – requirements, design, implementation, verification and release. During the design phase, the threat modeling methodology is used to apply a structured approach to possible threat scenarios which allows the identification of security vulnerabilities, determination of risks and helps build accurate security and mitigation measures [31].

The SDL's threat modeling process bases on the SRTIDE per element approach which in a simplified way consist of four parts. It starts by creating a system diagram using Data Flow Diagrams (DFDs) and then follows by identifying the threats with STRIDE. The third part is mitigating, which means addressing each threat to eliminate or alleviate the issue. Final part is validation, where the whole threat modeling process is validated against quality of threats, their mitigations and information captured from dependencies and assumptions [32].

A good practice for starting an effective threat modeling with SDL is following the steps proposed in "Introduction to Threat Modeling" slideshow [32], that are:

- Develop a draft threat model with Data Flow Diagrams.
- Start with DFD walkthrough.
- Identify most interesting elements like assets and entry points or trust boundaries.
- Use STRIDE against those interesting elements.

- Look at threats that cross elements or recur.

As threat modeling is a core element of the Microsoft Security Development Lifecycle, Microsoft has created the SDL Threat Modeling Tool which isn't designed only for security experts, but for all developers and software architects which gives instructions on creating and analyzing threat models. Additionally, the SDL Threat Modeling Tool differentiates itself from similar tools and methodologies in two main areas. Firstly, it is designed having developers in mind and centered on software, not like other threat modeling approaches that focus on assets or attackers. Secondly, this tool's approach to threat modeling is aimed on design analysis, rather than requirements analysis or the combination of two [33].

## 3.2 Process for Attack Simulation & Threat Analysis (PASTA)

The Process for Attack Simulation and Threat Analysis (PASTA) is a seven-step application threat modeling methodology created by Marco Morana and Tony UcedaVelez. This approach is meant to be applicable in majority of application development methodologies with the focus of addressing the most viable threats. The process integrates application threat analysis, application threat modeling methodology and risk or asset based approach. The seven stages build up to impact of threat to application and business [34].

The process starts with defining business objectives, security and compliance requirements, along with performing preliminary business impact analysis. The next step is defining the technical scope to identify boundaries and application dependencies form network, infrastructure and software. Third stage bases around the decomposition of the application by creating Data Flow Diagrams to identify all the actors, assets, entry points and other important factors. Forth step is threat analysis where the identification and extraction of threat information is used to get and overview of threat-attack scenarios used by the attackers. In the next steps, the vulnerabilities and weaknesses analysis of application security controls is followed by the attacks enumeration with modeling. In the final stage, risk and business impact analysis provides information about impacts, residual risks and necessary countermeasure development [35].

This process in overall in cooperates different threat modeling methodologies nicely, like the usage of attack trees to get an attacker-centric view of the threats with the inclusion of risk and impact analysis. For organizations, this kind of approach provides valuable asset-centric mitigation planning. Also, from the risk and business impact analysis perspective, this process goes far beyond any regular software development threat modeling, as it links the vulnerability management to key business decision makers, which ultimately can make a big difference in software development and security [34].

## 3.3 Trike

Trike is an open source threat modeling project and a tool which was created in 2006 with the focus to improve the existing threat modeling approaches [36]. The methodology has three versions, but unfortunately only the first is fully documented and somewhat usable. As such, according to the first version of Trike, it is said to differentiate itself from other threat modeling methodologies by three key areas. Firstly, the possibility of high levels of automation within the system, which allows quicker and better result. Secondly, the defensive perspective, which addresses all the weaknesses and attacks. And finally, the degree of formalism, which supports automation within the Trike methodology [37].

In overall, the process is similar to many others, as it includes identifying model requirements, like actors, assets and actions which are implemented within Data Flow Diagrams and Use Flows. Threat determination is done by actor-asset-action matrix and its followed by modeling of attacks, attack trees and attack graphs which lead to weaknesses and vulnerabilities. Mitigations and attack libraries are used in the end of the threat modeling framework. Next and final phase is risk modeling, where all the weaknesses and vulnerabilities probabilities are analyzed [37].

From the looks of it, Trike remains still in an experimental stage with its inadequate or partial documentation and support. Moreover, the development of Trike seems to have been stopped, as the official website's last updates date back to the year 2012 [36] and the final update on their threat modeling tool was in April 2013 [38]. This definitely makes Trike as a difficult choice to implement and understand in the long run.

## 3.4 OWASP Application Threat Modeling (ATM)

The OWASP Application Threat Modeling is a structured approach for analyzing security of an application by identifying, quantifying and addressing security risks. It's not a code reviewing methodology, but rather a process in the Software Development Life Cycle (SDLC) that ensures that software is developed following the security mindset from the beginning. Moreover, when source code analysis happens to be outside the SDLC, like with already created applications, then the threat modeling results can simplify the code analysis. This is done by prioritizing security code review components according to high risk threats, in comparison to looking at all components equally [39].

The OWASP threat modeling process is divided into three high level phases. The process begins with decomposing the application, which means getting to know the application and how it can be used. During this phase, the entry points, assets and trust levels are identified and put to use in producing Data Flow Diagrams. The next step is determining the potential threats from the attacker side using STRIDE methodology or the defensive perspective categorization called the Application Security Frame (ASF). For threats visual representations and illustrations, the use and abuse cases are created. They show how existing security measures are bypassed or where they are non-existent. The ranking of threats is done either by DREAD or Generic Risk Model, which bases on general risk factors like likelihood and impact. The final phase is all about countermeasures and mitigation strategies where countermeasures can be determined with the help from threat-countermeasure mapping lists and mitigations are done according on threat priority or business impact [39].

In overall, OWASP's approach is very similar to Microsoft's SDL as it has the same building blocks, but it differentiates itself with added threat ranking models and visual representations by use and abuse cases. Even with these additions, it seems to be quite straight-forward and simple to implement, whether the needs are for an attacker- or defender-centric analysis. Moreover, it seems to have potential to be applicable outside the context of an application security analyzation.

## 3.5 Comparative analysis

All these most commonly known methodologies focus on threat modeling, but they all do it differently. Table 1 compares these approaches by their key principles and concludes their suitability in terms of complexity and place to implement**.**

Table 1. Comparison of threat modeling approaches.

| | **Microsoft SDL** | **PASTA** | **Trike v1** | **OWASP ATM** |
|---|---|---|---|---|
| **Stages in threat model** | Four | Seven | Four | Three |
| **Perspective of threats** | Software-centric | Attacker-centric | Defender-centric | Attacker or defender-centric |
| **Analysis focus** | Design | Requirements | Requirements | Design |
| **System modeling methodology** | Data Flow Diagrams | Data Flow Diagrams | Data Flow Diagrams & Use Flows | Data Flow Diagrams |
| **Threat determination methodology** | STRIDE | Threat-attack scenarios | Actor-asset-action matrix | STRIDE or ASF |
| **Threat/risk ranking methodology** | None (all threats are addressed equally) | Common Vulnerability Scoring System or Common Weakness Scoring System | Trike's proprietary and experimental approach | DREAD or Generic Risk Model |
| **Threats visual modeling methodology** | None | Attack Trees, Use and Abuse or Misuse Cases | Attacks, Attack Trees, Attack Graphs | Use and Abuse or Misuse Cases |
| **Complexity (Low, Medium, High)** | Medium | High | High | Medium |
| **Documentation and support (Poor, Good, Very Good)** | Very good | Very good | Poor | Good |

| | Microsoft SDL | PASTA | Trike v1 | OWASP ATM |
|---|---|---|---|---|
| **Conclusion** | A simple threat modeling for any Software Development Life Cycle with its documentation and own threat modeling tool. Non-existent ranking and visualization of threats can be a problem for some. | An advanced solution that focuses mainly on business impact analysis regards to threats. Suitable for medium or large-sized businesses with their own dedicated teams for threat analysis, due to the complexity and large scale of the approach. | An experimental and complex solution with poor documentation, but with some levels of automation by its platform independent tool. Suitable for testing purposes, but not for businesses or long term implementations. | A solution that is suitable for many different scenarios with its optional threat determination and ranking approaches, when still remaining simple to implement. Lack of automatization and tools make it rather time consuming to implement. |

The comparative analysis clearly differentiates all four threat modeling methodologies and the concluding section of the Table 1 describes their suitability in different contexts. As a result, it can be seen that there currently isn't a solution that fully meets the needs of the proposed conceptual framework. Given the need for overall simplicity and repeatability in the suitable framework, the closest usable approach is the combination of Microsoft's SDL threat modeling and OWASP's Application Threat Modeling solutions.

# 4 Methodology

Now, when knowing the background of the eighth generation home gaming consoles and related works in the field of threat modeling, the author of this thesis introduces a new proof-of-concept threat modeling framework called Home Gaming Consoles Threat Modeling (HGCTM). This provides a solution for determining, rating, prioritizing and modeling test cases of potential threats in regards to home gaming consoles. At the same time, following a certain structure that can be easily repeated. This chapter will first go over the fundamental threat modeling principles, which are then followed by the chosen methodologies for HGCTM framework.

## 4.1 Fundamental threat modeling principles

The fundamental principles of threat modeling can be looked as a four step framework that consists of modeling the system, finding the threats, addressing the threats and validating. These principles are mainly used within the software development life cycle, but as with many different approaches and methodologies, there is more than one way to do threat modelling. That is why, the right approach is the one that finds good threats in regards to one's specific needs and views. In overall, threat modeling can be looked like a security focused version control [40].

### 4.1.1 System modeling

The most common way to start threat modeling is to model the system, like it is seen from the previous chapter. From the defenders' perspective, it is done when system is being built, deployed or changed. From the adversarial perspective however, when its already built or deployed. Using diagramming like Data Flow Diagrams for modeling is a good way to represent how data flows through the system, as focus is on the data flow, not control flow. The diagram tells a story about the process that moves the data and who uses it. In overall, the diagram doesn't include components that aren't used or are described in such detail that makes the whole diagram hard to read [40].

Data Flow Diagrams (DFDs) are already known from the 1970s, as they were popular in software development. Currently there are two common methods for data flow diagrams, but they have different visual representations for processes, data stores, data flow and

external entities. The first, Yourdon and Coad type, DFDs are known to be used for system analysis and design, whereas the second, Gane and Sarson type, is commonly seen in information systems visualizations. The most notable visual difference between the two types is the representation of the processes. For the first, processes are drawn as circles, while in the other they are squares with rounded corners [41]. As the analysis focus in the proposed threat modeling framework is on system design, the data flow diagrams are created according to Yourdon and Coad's type.

### 4.1.2 Determining threats

The next step is finding the potential threats within the system model by applying threat determination approaches. Good practice is to start with external entities or use cases that drive activity. Moreover, never ignore a threat because it isn't currently in the determination category, as later it might not come up again. Also, the focus is on feasible threats like exploiting vulnerabilities that are left unpatched, instead of hoping that there's a backdoor implemented at the chip factory [40, pp. 11-12]**.** Some of the common methodologies for finding and categorizing threats are STRIDE, ASF or custom solutions created according to approach's needs.

The STRIDE threat classification and identification method uses six threat categories which also make up the acronym. It consists of the attacker's goals such as Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege [42]. The spoofing the identity action is aimed to illegally access or use another user's credentials. The tampering with data is all about maliciously changing or modifying data, like sensitive data in a database or altering data that is in transit between two systems over a network. The repudiation goal is to perform illegal actions within the system that doesn't have the ability to trace the alterations. The information disclosure is the attacker's goal to be able to read the information that one was not given access or the possibility to read unauthorized data in transit. The denial of service aims to deny access of legitimate users to the systems and services by making them unavailable or unreliable. The final, elevation of privilege goal for an attacker is to elevate one's privileges to system's resources by having access to required unauthorized information or being able to compromise the system [43].

The Application Security Frame (ASF) classification proposed by OWASP's Application Threat Modeling [39], consists of following threat categories:

- Auditing & Logging,
- Authentication,
- Authorization,
- Configuration Management,
- Data Protection in Storage and Transit,
- Data Validation,
- Exception Management.

The goal of the ASF is to help identify threats from the defensive perspective and focus on certain weaknesses like vulnerabilities in security controls. Commonly, the identification process is done by threat-lists and examples until all the possible threats in the list are assessed according to each component [44].

### 4.1.3 Addressing threats

Third stage is addressing the found threats with threat managing or ranking methodologies. From the defenders' perspective, there are four types of managing actions that are taken against threats – mitigate, eliminate, transfer or accept. Mitigation is about adding ways to make it harder for threat to realize. Whereas elimination is majorly achieved by eliminating system's features completely. Transferring threats is about moving the responsibility to someone or something other to handle the risk – like passing the system's authentication to the operating system or external authentication server. The acceptance of risk is all about accepting the threats that are either too unlikely or too expensive to address in any other means [40, pp. 12-13].

From the adversarial perspective however, there isn't anything to manage as the system is already built or deployed. Thus, the only reasonable action is ranking and prioritizing the found threats. This is done to either understand the threats severity better or prioritize the testing of potential threats. There are many different ways to rank threats and vulnerabilities, but they are mostly done through some of the following methodologies.

The DREAD model is used to calculate risk rating for a threat by following the five principles, which first letters of each category also form the acronym. The first, Damage Potential is all about the damage when the vulnerability is exploited by an attacker. The Reproducibility looks at how quickly can an attacker reproduce the attack, while Exploitability wants to know how simple is the exploitation. Affected Users metric needs

an approximate percentage about the affected users. And the final part, Discoverability, looks on how discoverable is the vulnerability for the attacker. For each principle, a simple scheme such as High (1), Medium (2), and Low (3) is used to calculate the final value [45].

The OWASP Risk Rating Methodology is a risk severity estimation and ranking approach which bases on standard methodologies, but is made for application security. It follows the generic Likelihood multiplied Impact approach, where assessing the overall likelihood and impact in terms of high, medium, or low is sufficient. In more detail, the likelihood estimates multiple Threat Agent Factors as well as Vulnerability Factors. From the impact side, the Technical Impact Factors and Business Impact Factors are also estimated. Even though the Business Impact is considered more important, when lacking such information, Technical Impact factors are acceptable as well. The final severity ranking of low, medium or high is determined by the Impact multiplied Likelihood matrix [46].

The Common Vulnerability Scoring System (CVSS) is a universal rating framework focusing on IT related vulnerabilities, which is owned by US-based non-profit organization. The vulnerabilities severity rating is done by three metric groups: Base, Temporal and Environmental. The Base group present the main details about the weakness, the Temporal metric displays the symptoms that are changing in time, and the Environmental group is about vulnerability attributes relating to user's particular environment. The Base metrics produces a score from 0 to 10, where 10 being the most severe, which can be then modified by rating the Temporal and Environmental metrics [47].

The Common Weakness Scoring System (CWSS) is an open community-based system, which rates and prioritizes software weaknesses in three metric groups: Base Finding, Attack Surface and Environmental. Similarly, to CVSS, the Base Finding metric group is about the essential risk and details of the weakness. The Attack Surface is about the characteristics that an attacker requires and meets in order to exploit the weakness. Finally, the Environmental symptoms are unique to a particular context. Additionally, every group has their own multiple factors, that are used to compute the overall CWSS score for a weakness. In the end, the CWSS process produces a score between 0 and 100 [48].

### 4.1.4 Validating or visual modeling

For the most part, the final phase of threat modeling is validating the whole work for completeness and effectiveness. From the defenders' side, it is done by testing the threat mitigations, checking the code, quality assuring the threat model or processing aspects of threat addressment. Validation of individual threats is done by checking that the right action is taken in terms of each threat and that all intended threats are found [40, pp. 189-202].

From the attacker's perspective, the final approach would be to plan for testing the threats. As such, it is wise to model test cases out of the highest rated or most interesting threats. Those test cases can be then tested against potential vulnerabilities with different tools. The results of these tests either validate or invalidate the potential threats found with threat modeling process. In the next iteration of this threat model, the corrections to the created DFDs can be done according to previous testing results.

## 4.2 The methodology of the HGCTM

The proposed proof-of-concept Home Gaming Consoles Threat Modeling (HGCTM) framework follows the four main threat modeling principles described in the previous sections, but with the approach centered on adversarial perspective and on systems that have already been built or deployed. The focus of the analysis is system design and feature based, because home gaming consoles are closed systems (black boxes) with no publically available technical or design documentation. Moreover, the scope is according to eighth generation gaming consoles and their network based features. The exact methodology chosen for the proposed framework and tools for each phase is described below.

### 4.2.1 Modeling the system

Form the system modeling approach, the HGCTM will follow the Data Flow Diagrams (DFD) methodology, where the flow of data in the system is described. This approach is taken due to its simplicity and applicability in the other contexts of threat modeling. As seen from the related works, it is used by all compared threat modeling approaches and that is why, it's easily recognized by other threat modelers.

Another reason is also the availability of already developed and free tool called Microsoft's Threat Modeling Tool (TMT), which is rather easy to use and provides some level of automatization. Even though the Microsoft's Threat Modeling Tool isn't centered on assets or attackers, the threat analyzing of system design is still relevant. As such, HGCTM will be using the latest version 2016 of the Microsoft Threat Modeling Tool and its well documented guide [49].

Due to the fact that home gaming consoles are closed systems with no publically available detailed security documentation, the only way to gain real evidential information about these systems security is via practical use cases, forensic analysis or penetration testing. Moreover, the author of this thesis had the chance to query one of the Microsoft's senior software engineers working in Xbox Research and Development team about the possible documentation sources for security measures of the Xbox. The answer was as expected – the technical aspects of the security are not public and he has no liberty to share how the system works or what security counter measures are in place.

As such, the first iterations of HGCTM system modeling data flow diagrams will be based on data collected from practical use cases and other researcher's forensic evidences. The following iterations can build upon already created models and improve the initial ones by validating or invalidating the found threats and security measures.

### 4.2.2 Determining the threats

For determining and categorizing the threats in Home Gaming Consoles Threat Modeling framework, the STRIDE methodology is used. It is suitable, because it has the attacker's point of view and covers all the main threat categories in a structured way. Also, like with DFDs in system modeling, the STRIDE methodology is known from the related works in the field and no new documentation is needed to be created. One way for determining threats using STRIDE is the gamification approach through the Elevation of Privilege threat modeling game developed by Adam Shostack [40, pp. 206-208]. It is a good idea, but determining threats in the context of speed, something automated is needed.

Fortunately, the previously chosen system modeling tool, the Microsoft's Threat Modeling Tool, also supports the STRIDE per element categorization. After completing the DFD model with TMT and switching over to analysis view, the list of automatically generated threats is displayed. Then, choosing the category filter with Spoofing,

Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privileges – the complete list of STRIDE determined threats are shown.

As the quality and the quantity of these automatically created threats depend on the accuracy of the modeled system's DFD and the used template – the first iterations will have quantities of threats, rather than fewer quality threats. Luckily, the Threat Modeling Tool has the ability to use custom made templates with predefined stencils and threats lists [50]. This provides an excellent possibility for the HGCTM framework to have its own specifically configured custom template.

### 4.2.3 Addressing the threats

In the addressing the threats phase, each threat is looked at within the Threat Modeling Tool and a certain state appointed to them. State can be either "Needs Investigating", "Not Applicable" or "Mitigated." The not applicable threats are either removed from the list or kept with the given status if the modeler is uncertain or knows that the same DFD will be reused in the future. The threats that are known to be mitigated in the previous software versions of gaming consoles are set as mitigated, because it is unlikely that already patched vulnerabilities pose a threat.

As the first iterations of the HGCTM base their knowledge merely on the information collected from practical use cases and other researcher's forensic evidences, the list of threats is quite long and time consuming to address. Fortunately, with each model's iteration and updated version of HGCTM's template, the threats list shortens and the quality of threats improve. Moreover, as Threat Modeling Tool 2016 supports applying improved templates over already modeled DFDs, it saves a lot of time by improving already existing models [50].

After addressing the threats according to state, the ones with "Need Investigating" are ranked according to the OWASP Risk Rating Methodology. Firstly, this model is used, because it already has semi-automatic Excel based tool for easy ranking [51]. Secondly, this ranking method is one of the simplest to understand and implement, because the level of risk is determined in combination of threat's likelihood and the impact. Moreover, as likelihood and impact have their additional factors, overall severity rating will be quite accurate. In the end, each remaining threat that needs an investigation, gets their priority level set to either low, medium or high [46].

### 4.2.4 Modeling test cases

In the modeling the test cases phase, the highest ranking or most the interesting potential threats are modeled using Misuse cases methodology. Misuse cases are used, because they extend the standard Unified Modeling Language (UML) use cases model's security concerns. This approach provides a clear understanding of the threat by defining misuser's sequence of steps. When these steps are successfully performed by an attacker, the stakeholders or the system is harmed as a result. Here, the misuser is considered as a threat agent and an attack method as misuse case. A threat is then modelled in combination of misuser and misuse case [52].

The following two tables are created and modified by the author of this thesis, to make Misuse cases modeling simpler from the Threat Modeling Tool's results. These tables are created using the information from OWASP's "STRIDE Threat List" and "Fundamentals of Secure System Modelling" unpublished draft by R. Matulevičius [43] [52]. Due to the fact that gaming consoles threat modeling is done from the adversarial perspective using STRIDE, but the Misuse cases need a defender's view of the affected security controls/criterions, the conversion is required. Table 2 is used for Security Criterion identification according to potential threat's type by STRIDE category.

Table 2. STRIDE Threat type to Security Criterion.

| **Threat type (STRIDE)** | **Security Criterion** |
|---|---|
| Spoofing | Authentication |
| Tampering | Integrity |
| Repudiation | Non-repudiation |
| Information Disclosure | Confidentiality |
| Denial of Service | Availability |
| Elevation of Privilege | Authorization |

To make the Misuse cases creation even more easier, the Table 3 helps to identify rest of the Misuse cases items from the Potential Threat's information. An example about one of the potential PlayStation 4 threats is included into the table for guidance. Also, it is noted, that the required security criterion is devised with the help from Table 2. After identifying all the necessary Misuse cases items, the modeling of test cases can begin following the guide and syntax.

Table 3. Misuse Cases according to Potential Threats.

| **Misuse Case item** | **According to Potential Threats** | **Potential Threat's example** |
|---|---|---|
| System boundary | Gaming Console<br>(PlayStation 4, Xbox One, Wii U) | PlayStation 4 |
| Use case | Action how valid user interacts with the Gaming Console or its feature. | User request Remote Play with companion device (Internet). |
| Vulnerability | Weakness or a flaw in Gaming Console's security. | Potential Lack of Input Validation for RemotePlay function. |
| Misuse case | Attack method. | Remote RP request may be tampered with by an attacker. |
| Impact | The potential negative consequence. | May lead to denial of service attack against RemotePlay function or an elevation of privilege attack against RemotePlay function or an information disclosure by RemotePlay function. |
| Security criterion | Taken from the Table 2: STRIDE Threat type to Security Criterion | Integrity |

For modeling the Misuse cases, any modeling tool supporting Unified Modeling Language (UML) can be used, depending on needs or availability. For the sake of the free trial period and platform neutrality, the Online Diagram and Flowchart Software called Gliffy [53], is used within the proof-of-concept phase of the Home Gaming Consoles Threat Modeling framework. The guide and syntax that is being used by the HGCTM to model the Misuse cases is taken from the Chapter 7 in "Fundamentals of Secure System Modelling" unpublished draft by R. Matulevičius [52]. As the used source is yet to be published, any available Abuse cases syntax is significant.

As testing of these modeled Misuse cases is out of the scope of this research paper, it really depends on the specific threat and modeler's skills how it is performed. Testing can be done using publically available penetration testing tools or for more advanced users, they create their own. The best way to validate these potential threats test cases is through a security testers or researcher's community, where the combined knowledge and skillset tests and confirms or denies the potential threats.

# 5 Implementation of the framework

In this chapter, the practical implementation of the HGCTM framework is carried out and explained with samples. The underlying implementation's test case is carried through on eighth generation gaming console called the PlayStation 4 (PS4) and its two network based features. PS4 is chosen over the Microsoft's Xbox One, because Microsoft is known for its Security Development Lifecycle policy [30]. More precisely, its threat modeling approach, where all the found threats are addressed during software development lifecycle [32]. It is fair to assume that this approach extends to Xbox One's software development and therefore is a harder candidate for this test case. Moreover, PlayStation 4 is chosen for a better test case candidate in the following reasons:

- most popular eighth generation home gaming console;
- more network based features compared to competitors;
- thesis author has a personal PS4 device.

In overall, the practical work consists of four phases. For successful implementation, the following tools, templates and documentations are used:

- The Microsoft Threat Modeling Tool (TMT) 2016 with documentation [49];
- Home Gaming Consoles Threat Modeling (HGCTM) TMT 2016 template;
- Excel template for OWASP Risk Rating Methodology [51];
- Misuse cases modeling tool depending on the needs or availability:
- Any modeling tool supporting Unified Modeling Language;
    - Online Diagram and Flowchart Software called Gliffy [53];
    - Table 2 and Table 3 of this thesis, for easier Misuse cases modeling;
- Misuse cases modeling guide and syntax from "Fundamentals of Secure System Modelling" unpublished draft by R. Matulevičius [52].

The resulting documentation includes the Threat Modeling Tool's Custom Report containing Data Flow Diagrams with the list of potential threats. Additionally, included are the created Misuse cases models and their descriptive tables, not only for highest ranking or most interesting threats, but for all threats in aim to provide supplementary samples. The completed report can be found in the Appendix 2.

## 5.1 Model the system

The first step in Home Gaming Consoles Threat Modeling framework is to model the system or parts of it. It starts with opening the Threat Modeling Tool 2016 and choosing the HGCTM template for new models. The template provides some custom stencils and improved list of STRIDE categorized threats. The template selection can be seen in the Figure 1. Then the "Create A Model" is chosen and the empty canvas is displayed. That is where the system modeling will take place. Before starting to model the Data Flow Diagrams of the system – the scope, assumptions, external dependencies and overall information of the threat model is defined.

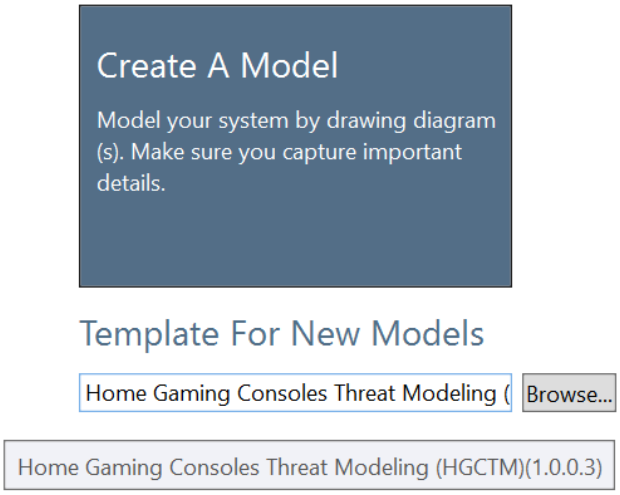



Figure 1. Choosing the template.

It is really important to define the system scope, assumptions, and external dependencies beforehand, because if the scope is too wide or the number of external dependencies large, the modeling of DFDs will get too complicated and hard to follow very quickly. As the DFDs are hierarchical, the models start at a very high and simple level with each following going deeper when more is known about the system or features.

The defining is done by choosing "File" and "Threat Model Information" within the TMT and filling in the details. The Figure 2 shows an example of PlayStation 4 Network Traffic's threat modeling information that takes only two network features into scope. The features called Second Screen and Remote Play are included, because the first one is meant to be used within the LAN, but the other is useable over the Internet as well. Assumptions regarding these features and use cases are made based on publically available PS4 user manuals [54] [55].

External dependencies are items external to the gaming consoles. In this example, Sony's authentication server is used by gaming console and companion application on the companion device to validate and authorize the PlayStation Network (PSN) accounts. Also, the companion device itself is an external dependency, because it connects to the gaming console externally. At the bottom of the information window, the title and the version number of the used template is filled in automatically. In this example, the HGCTM's template with version number 1.0.0.3 is used. As all these fields are simple textboxes and no customization currently available, the modeler's will follow the given sample's style seen in Figure 2.

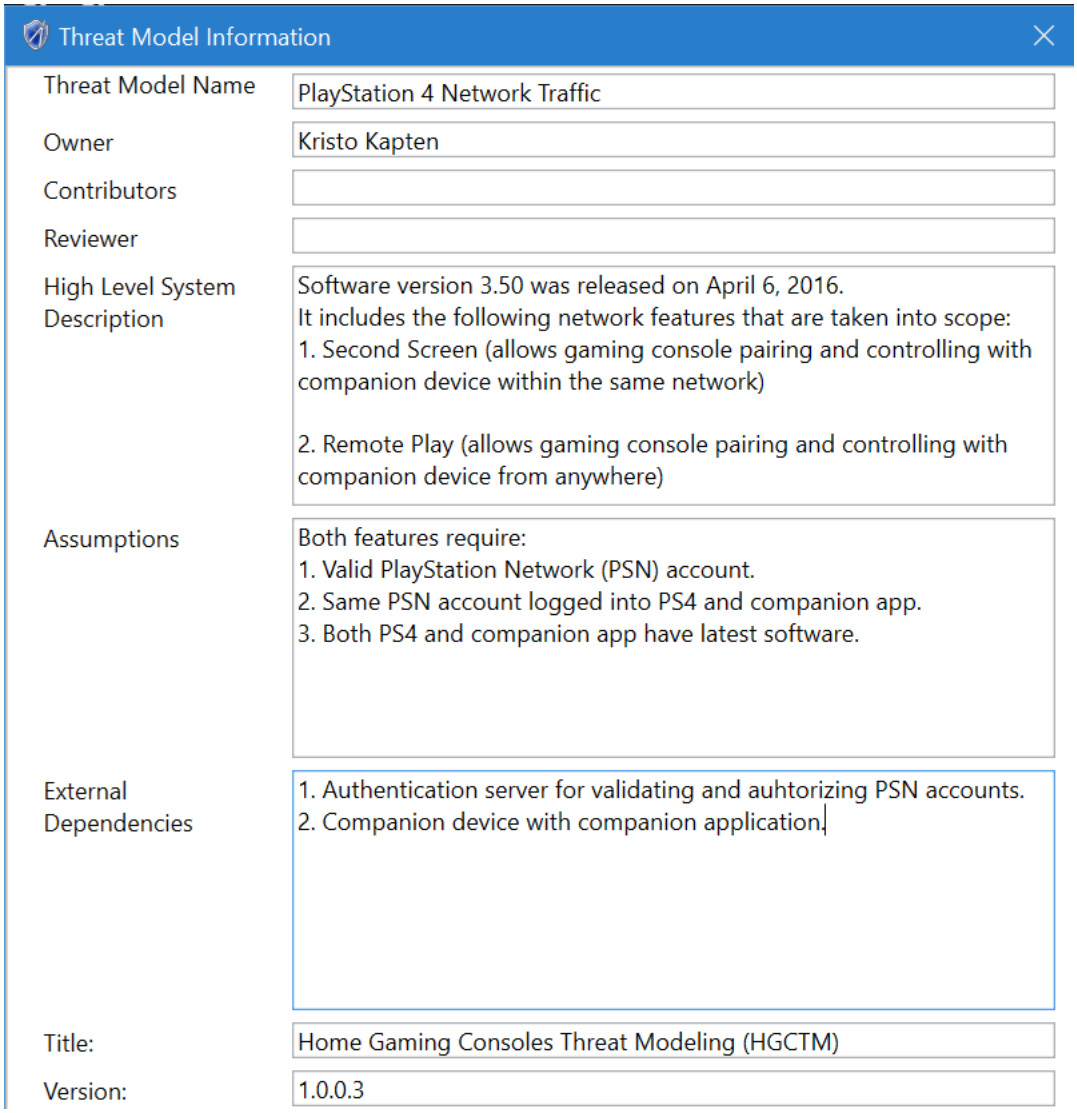



Figure 2. PlayStation 4 Network Traffic's Threat Modeling Information.

Next step, after closing the threat model's information window, is the modeling of DFDs with the help from Threat Modeling Tool's user guide and according to the details just defined [50]. The modeling is quite straight-forward, because the pre-defined stencils are

categorized and their titles rather self-explanatory. Anyone with the help from the Threat Modeling Tool's documentation is be able to create at least a high level DFD.

For more advanced modelers, it is possible to appoint different attributes to each stencil or connection. For example, if it is known that communication between the mobile device and gaming console is done via Bluetooth or Wi-Fi, it can be set accordingly. This not only gives an exact representation in terms of reality, but rises the quality of potential threats in the determination phase with the relating vulnerabilities defined within the framework's template. An example of the created PlayStation 4 network traffic's DFD is shown in the Figure 3.

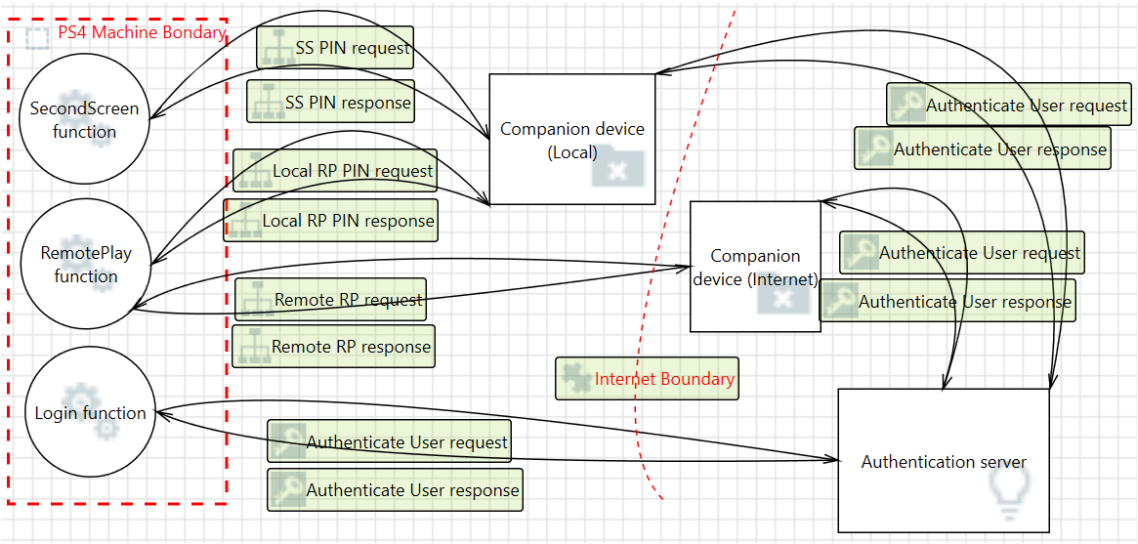



Figure 3. PlayStation 4 Network Traffic's DFD.

As it can be seen from the Figure 3, the created DFD has many items and connections between them. Most notably, there are two separate companion devices, because the device can be used to communicate with PlayStation 4 through the Remote Play function within the local area network or outside of it. To cover all possible interactions, both are modeled on to the DFD. Also, the communication links between the Authentication server and devices are marked as secure with the key icon, because they are generally known to use secure communication channels. Moreover, due to the two different companion devices and an external Authentication server, the Internet Boundary with red dotted line in the center of DFD is used to separate and distinguish different boundaries. All additional, but visually unseen configurations to the DFD originate from practical use cases or other researcher's forensic analysis.

## 5.2 Determine the threats

Next step is finding threats from the DFDs created in the previous phase. Fortunately, the Threat Modeling Tool does some of the work automatically. Switching over to "Analysis View" provides the list of automatically generated threats. Due to the fact that default template in TMT has more categories for threats than the chosen STRIDE methodology, the filtering is done accordingly. An example of applying the STRIDE filter is show in the Figure 4. Luckily, this filtering is done only on the conceptual iterations, because the fully configured HGCTM template removes the additional categories.

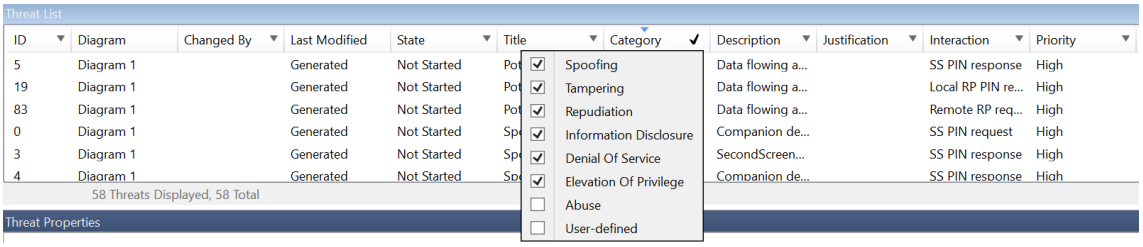



Figure 4. Threat List filtering with STRIDE.

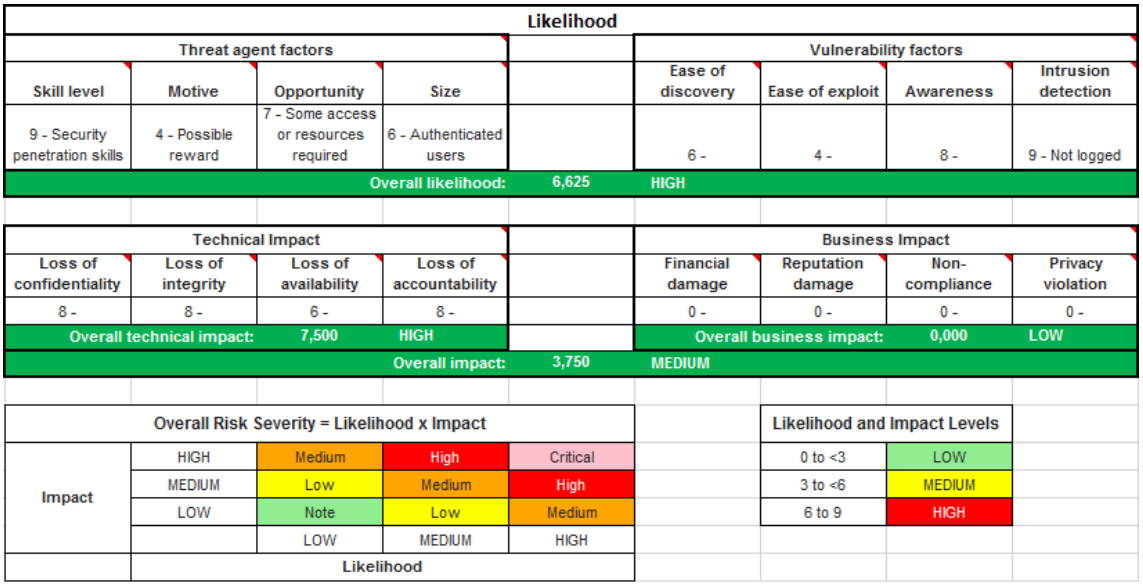
After the automatic determination and filtration, the list of threats is very long and overwhelming. This is due to the numerous items or connections within the DFD, as well as the over-generic threats that apply to all unspecified connections. Even though, the automatic approach is quick and simple, the manual validation and suitability of the threats is still required. Due to the fact that the next phase will addresses each threat individually anyway, the validation of this is done within that phase.

## 5.3 Address the threats

In the addressing the threats phase, each threat is given a state according to the researcher's knowledge and information gathered from forensic or previous test cases results. Chosen states are either "Needs Investigating", "Not Applicable" or "Mitigated." The not applicable ones are removed from the list or kept for safe keeping for any future updates to the DFD or template.

After addressing the threats according to the state, the remaining ones that are chosen for investigation are filtered from the list and ranked following the OWASP Risk Rating Methodology template [51]. Their priority level is then set to either low, medium or high – according to the severity level calculated by the semi-automatic template. An example of risk severity rating can be seen in the Figure 5, which finds the severity level for one

of the possible remote code execution threats allowing elevation of privilege through the Remote Play function.



| Likelihood | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Threat agent factors | | | | | Vulnerability factors | | | |
| Skill level | Motive | Opportunity | Size | | Ease of discovery | Ease of exploit | Awareness | Intrusion detection |
| 9 - Security penetration skills | 4 - Possible reward | 7 - Some access or resources required | 6 - Authenticated users | | 6 - | 4 - | 8 - | 9 - Not logged |
| Overall likelihood: | | | | 6,625 | HIGH | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Technical Impact | | | | | Business Impact | | | |
| Loss of confidentiality | Loss of integrity | Loss of availability | Loss of accountability | | Financial damage | Reputation damage | Non-compliance | Privacy violation |
| 8 - | 8 - | 6 - | 8 - | | 0 - | 0 - | 0 - | 0 - |
| Overall technical impact: | | 7,500 | HIGH | | Overall business impact: | | 0,000 | LOW |
| Overall impact: | | | | 3,750 | MEDIUM | | | |

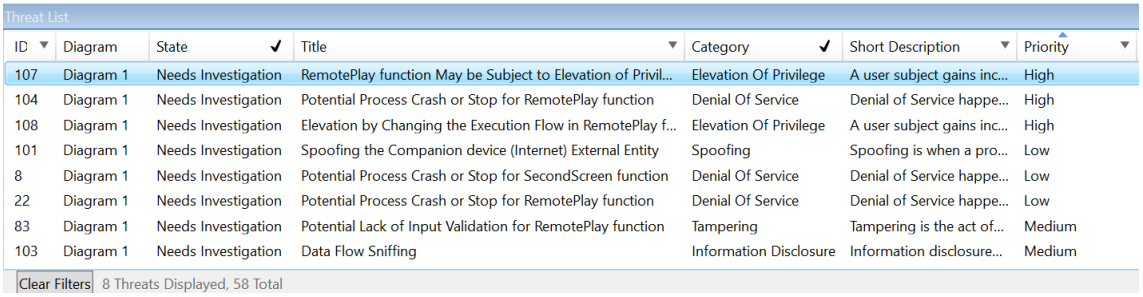
| Overall Risk Severity = Likelihood x Impact | | | | |
|---|---|---|---|---|
| Impact | HIGH | Medium | High | Critical |
| | MEDIUM | Low | Medium | High |
| | LOW | Note | Low | Medium |
| | | LOW | MEDIUM | HIGH |
| | Likelihood | | | |

| Likelihood and Impact Levels | |
|---|---|
| 0 to <3 | LOW |
| 3 to <6 | MEDIUM |
| 6 to 9 | HIGH |

Figure 5. Risk severity rating.

In the current risk rating example in Figure 5, the threat agent factors are set rather high, because this kind of remote attack requires real penetration knowledge from the attacker. Moreover, as this potential threat was discovered quite easily, the combination of these factors result in high level of risk likelihood. From the Business Impact factors, it can be seen that they are left undefined, because the real impact to Sony is completely unknown. As such, the next best are the Technical Impact factors, which are assessed in relation to PlayStation 4's gaming console and provide the joined impact level as medium. Now, having the likelihood high and impact medium, the overall risk severity rating according to given matrix results in high, which is also the priority for this threat.

Same rating and prioritization methodology is used also for the remaining threats in the list. An example of the final threat list after appointed states, rankings and filtering by needs investigation, is shown in the Figure 6.



Threat List

| ID | Diagram | State | Title | Category | Short Description | Priority |
|---|---|---|---|---|---|---|
| 107 | Diagram 1 | Needs Investigation | RemotePlay function May be Subject to Elevation of Privil... | Elevation Of Privilege | A user subject gains inc... | High |
| 104 | Diagram 1 | Needs Investigation | Potential Process Crash or Stop for RemotePlay function | Denial Of Service | Denial of Service happe... | High |
| 108 | Diagram 1 | Needs Investigation | Elevation by Changing the Execution Flow in RemotePlay f... | Elevation Of Privilege | A user subject gains inc... | High |
| 101 | Diagram 1 | Needs Investigation | Spoofing the Companion device (Internet) External Entity | Spoofing | Spoofing is when a pro... | Low |
| 8 | Diagram 1 | Needs Investigation | Potential Process Crash or Stop for SecondScreen function | Denial Of Service | Denial of Service happe... | Low |
| 22 | Diagram 1 | Needs Investigation | Potential Process Crash or Stop for RemotePlay function | Denial Of Service | Denial of Service happe... | Low |
| 83 | Diagram 1 | Needs Investigation | Potential Lack of Input Validation for RemotePlay function | Tampering | Tampering is the act of... | Medium |
| 103 | Diagram 1 | Needs Investigation | Data Flow Sniffing | Information Disclosure | Information disclosure... | Medium |

Clear Filters | 8 Threats Displayed, 58 Total

Figure 6. The final threat list.

As seen from the sample, the result of addressing and prioritizing all the determined threats, only 8 out of initial 58 remain. From these remaining eight threats, three have high, two medium and three low severity levels. Now it is up to the modeler to decide whether only the highest ranking or most interesting threats are modeled.

Unfortunately, as the modeling of test cases is outside of the Threat Modeling Tool's capabilities, it must be done separately. To continue, a custom report of the completed work is done by choosing "Reports" and "Generate Custom Report." From the custom report section, the threats that still need investigation is chosen and report generated as seen in the Figure 7.

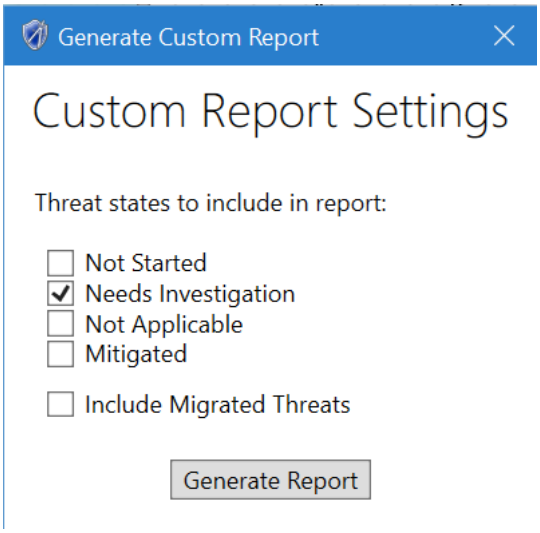



Figure 7. Custom Threat Modeling Tool's report.

The resulting custom generated report of the implementation is not yet complete report, because there is one more phase to the Home Gaming Consoles Threat Modeling framework. The work done in the final stage, test cases modeling, is manually added to the report afterwards.

## 5.4 Modeling the test cases

The final phase of the HGCTM framework is modeling the test cases. This is where the highest priority or the most interesting threats are modeled using Misuse cases. For modeling, any Unified Modeling Language tool can be used. For the sake of free trial period and platform neutrality, the Online Diagram and Flowchart Software called Gliffy is used to model the Misuse cases [48].

To supplement each Misuse case, a descriptive table of information is created. The table is filled in beforehand to make the overall Misuse case creation easier and provide textual representation of the Misuse case. This is done according to Table 2 and Table 3 from the section 4.2.4. of this thesis. The Table 2 is used for Security Criterion identification according to potential threat's type by STRIDE category. The Table 3 helps to identify rest of the Misuse case's items from the Potential Threat's information.

For the sake of better comprehension, the highest priority threat isn't taken as an example, but one of the simplest to model and understand, like the potential Denial of Service threat with Low severity level. The Table 4 is an example of the created descriptive table.

Table 4. Denial of Service Misuse Case descriptive table.

| **Misuse Case item** | **Potential Threat** |
|---|---|
| System boundary | PlayStation 4 |
| Use case | User request Remote Play with companion device. |
| Vulnerability | Process or a datastore is not able to service incoming requests or perform up to spec. |
| Misuse case | Denial of Service against Remote Play function. |
| Impact | Potential Process Crash or Stop for Remote Play function. |
| Security criterion | Availability |

After filling in the descriptive table, the modeling is done according to the table and Misuse Cases syntax guide. The created Denial of Service Misuse Case's sample model is displayed on the Figure 8.

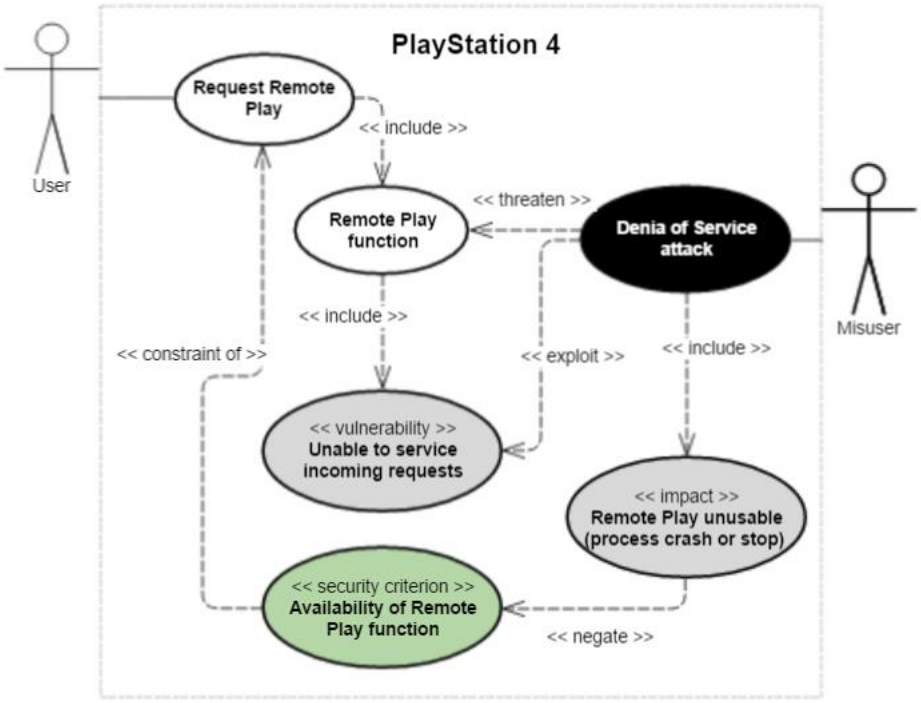



Figure 8. Denial of Service Misuse Case.

This concludes the final phase of the Home Gaming Consoles Threat Modeling framework's implementation. For the sake of the implementation report's completeness, the created Misuse Cases descriptive tables and models are manually added to the custom generated report created in the end of previous, addressing the threats, stage. Moreover, to provide additional samples for the conceptual work, all of the remaining threats are also modeled into Misuse cases with their respective tables and included to the report. The final and complete report can be seen in the Appendix 2.

# 6 Analysis of the framework

This chapter analyses the applicability and usability of the proposed framework in terms of goals and needs. Also, the evaluation of possible limitations and issues bases on the practical work done in the previous chapter and goals set for the Home Gaming Consoles Threat Modeling (HGCTM) framework. The final part compares the framework to experimental approaches used by other researchers and describes their shortcomings.

## 6.1 Applicability and usability

After implementing the HGCTM framework on PlayStation 4 and its network based features called Second Screen and Remote Play, it is evident, that the concept has potential. The main goal set for this proof of concept was to determine potential threats and deduct test cases for back box testing in regards to eighth generation gaming consoles and their network features. The practical approach and resulting documentation of the test case proves the viability of the work. Proving is done by listing the potential threats while having them ranked according to the chosen methodology. Ultimately solving the main goal set for HGCTM framework.

The second objective for this thesis was the adversarial perspective and the structured approach to cover all threat areas. In addition, the structured approach had to be repeatable. The attacker-centric view while covering all of the threat areas was solved by implementing the STRIDE per element threat determination and categorization methodology. Which was fortunately automated by the Threat Modeling Tool 2016 to some extent in the implementation phase. As a result of the automatization and custom template applicability process, the determination of threats according to HGCTM configuration is structured and easily repeatable. Thus, solving the second applicability criterion.

Third and final requirement was the overall simplicity of the proposed implementation, so that any cybersecurity testers or researchers can implement it easily. Currently, as the majority of the implementation is done manually and the final phase is outside the Threat Modeling Tool's capabilities, it might seem complicated. In reality however, a good supporting documentation is available for most phases and with further developments to the HGCTM template, the quantity of threats reduces and quality rises. Moreover, the

resulting thesis with the implementation guide and samples, proves overall easiness in usability.

Even though the applicability is mainly focused on eighth generation gaming consoles and their network based threats, it can also be suitable for previous or future generations. Also, with the right methodologies implemented, the focus of framework can be expanded beyond network level, as there are other ways to threaten system's security.

## 6.2 Limitations and possible issues

The first and foremost biggest limitation that became evident during the implementation and the system's modeling phase was the need for custom configured stencils. The custom stencils would provide appropriate icons to represent gaming console's system components and improve the speed for searching the right components. Moreover, the addition with gaming console related categorizations and stencil based technical configurations – the DFDs would reflect the reality more precisely. Additionally, this would allow for deeper and more precise levels of DFD modeling.

The second issue that can be fixed over time, but is currently quite time consuming to address, is the multitude of automatically generated STRIDE per element threats. With it, the appropriateness of each found threats is checked manually. This issue can be smoothened out with every new version of HGCTM's template, which will include custom stencils according to framework and STRIDE based threats that match the systems configurations.

The third limitation is the transitioning smoothness from determined, addressed and prioritized threats to creating test cases with Misuse cases. This issue is mainly due to the difference in terminology and undefined aspects of threats needed for Misuse cases modeling. The transitioning problem can be improved by detailing the STRIDE threats within the HGCTM template according to Misuse cases terminology.

One additional issue that arises in terms of Misuse cases creation is the fact that the guide and syntax used in HGCTM is yet to be published. The author of this thesis hopes that when this framework becomes more than just a proof-of-concept, the used guide is published in full. Moreover, as there are variety of samples within the thesis, it is feasible to model Misuse cases without the original document in hand.

Finally, the somewhat inconsiderable, but nice to have improvement to the whole concept would be to have references to tools and methods for testing the created test cases. Even though, the testing is out of the scope of this thesis and upon the modeler's skillset and knowledge, the less experienced testers or researchers could use some final guidance. Currently, the testing of all potential threats and test cases relies upon the modeler's knowledge.

## 6.3 Comparison to other approaches

Researchers, testers or others who have looked into gaming consoles security are using different approaches and methodologies to test these systems, a simple comparison between the proposed HGCTM framework and used approaches is provided. Due to the lack of academic papers regarding eighth generation gaming consoles security, the comparison is conducted between previous generation approaches. As such, two individual and experimental approaches conducted by Halvar Myrmo [12] and Walter W. Ridgewell [11] are compared against HGCTM framework in the Table 5.

Table 5. HGCTM framework compared to experimental approaches.

| | **Experimental approach 1** | **Experimental approach 2** | **Proposed HGCTM framework** |
|---|---|---|---|
| **Purpose** | Determine vulnerabilities. | Determine the state of vulnerabilities. | Determine potential threats and create test cases for black box testing. |
| **Scope** | Sixth and seventh generation gaming consoles and handheld devices. | Seventh generation gaming consoles and handheld devices. | Eighth generation home gaming consoles. (possibly suitable for others) |
| **Analysis focus** | Quantity | Quantity | Quality |
| **Methodology** | Use open source scanning and penetration testing tools. | Use open source scanning and penetration testing tools. | Design the system, determine potential threats, prioritize and create test cases. |
| **Structured** | No | Somewhat | Yes |
| **Repeatable** | No | No | Yes |
| **Threat areas covered** | Few (STRIDE) | Four (STRIDE) | All areas (STRIDE) |
| **Automated** | No | No | Partly |

As seen from the Table 5, all three approaches have the ultimate purpose of determining software weaknesses in gaming consoles, but the main difference lays in the methodology. While the experimental approaches rely mostly on open source scanning and testing tools, the HGCTM framework focuses on system design and threats analysis, which in turn produces viable test cases. This is also the reason why the experimental approaches focus on quantifying analysis of different devices, instead of doing qualitative analysis on a single system.

At first, it may seem easier and quicker to perform systems testing quantitatively without the use of any guide or framework, but in reality, doing so has many shortcomings. First, the most evident is the structural part, which is completely missing for the first approach, but somewhat implemented in the other. The somewhat structured approach means that it uses some elements to narrow down and track the testing of gaming consoles, but not fully. Secondly, as structure is essential for reproducing the exact test cases in different devices or software versions, they also lack in the repeatability possibility.

In comparison, the HGCTM framework uses structured approach, which not only helps determine most viable weaknesses in the system, but provides easy repeatability for different systems or versions. Even though, the implementation of HGCTM itself takes time, it saves a lot in scanning and testing phases, as only potential threat areas are tested. This also became evident from the Halvar Myrmo's research, where scanning whole ranges of ports required too much time and produced questionable results [12, p. 26].

Additional deficiencies in experimental approaches, can be seen in the threat areas covered and the automation section of the table. Due to the fact that first approach has no real structure to follow, only few threat areas are tested during the research. For the second approach however, it was able to cover majority of STRIDE's categories thanks to the partial structure, while missing only the Repudiation and Elevation of Privilege. As HGCTM framework uses STRIDE by default, it leaves no chance for test cases to miss potential threat areas or focus too much energy on irrelevant or already mitigated areas. In the automated section, the HGCTM is the only approach doing at least some parts automatically, even if potential threats are validated manually and rating is done semi-automatically.

Even though, the final penetration testing is outside of the HGCTM's scope, it provides necessary structure and assurances for repeatability and threat coverage, when used in similar experiments. This in overall allows for future improvements and collaboration on similar test cases, which is not feasible for compared approaches. To conclude the comparative analysis of experimental approaches to the HGCTM framework, it is safe to say that the proposed conceptual framework proves its superiority.

# 7 Conclusions and future works

This thesis focuses on determining network based threats and creating black box test cases in regards to eighth generation home gaming consoles. More precisely, it researches the viability of creating a proof of concept approach for covering all threat areas in a structured and repeatable manner. The selected methodology consists of the theoretical analysis of different threat modeling approaches, their key principles, as well as the practical work on creating and testing the concept of Home Gaming Consoles Threat Modeling (HGCTM) framework.

The background chapter of this thesis gives an overview of the eighth generation gaming consoles and their most known network features. Furthermore, the preliminary forensic analysis on two most popular gaming consoles are reviewed for system's security measures.

After having reviewed the systems and gained some knowledge about their security measures, the author analyses most common threat modeling approaches and their methodologies. The comparison of existing solutions concludes that even though each approach is unique, none of them are fully compatible to meet the needs and requirements set for gaming consoles threat modeling.

To solve the given problem, the author proposes a concept called Home Gaming Consoles Threat Modeling (HGCTM) framework. It follows the four key principles of threat modeling – modeling the system, determining, addressing and modeling the threats, with the methodology for each principle chosen according to specific goals and needs.

For validating the practicality of the chosen methodologies and overall proposed framework, the implementation chapter covers key principles in terms of practical work. The implementation's test case is done on the most popular eighth generation gaming console – the PlayStation 4. The complete threat modeling report containing the output of the framework can be found in the Appendix 2.

In the analysis phase of the thesis, applicability and usability of the proposed framework according to goals and needs are evaluated. The initial results, which bases on the practical work done in the implementation section, prove the viability of the concept, even

though some limitations were discovered. In the end, a comparative analysis between experimental approaches and the HGCTM framework further proves its usefulness.

The overall results of this thesis are important, because it shows that when existing approaches are not suitable, it is possible to present a new one by following the key principles and choosing the methodologies according to goals and needs. Moreover, the implementation of the conceptual framework on PlayStation 4's network features resulted in potential threats, which rises the importance of home gaming consoles security testing.

As the HGCTM framework is still in its early stages, future developments for improvements are evident. The most notable developments will be made on the Threat Modeling Tool's HGCTM template and creation of test cases. First, the template will include already configured stencils relating to eighth generation gaming consoles. Also, in relation to custom template, known threats are included into the STRIDE categories and cleared from unrelated ones. Second, the creation of the Misuse cases will be more seamless in terms of threats results.

Finally, it is paramount to realize that even though the HGCTM framework is introduced to be a solution for determining home gaming consoles threats in a structured and repeatable manner, and to create models for the basis of testing – it is still a proof-of-concept and not yet fully ready for everyday implementations.

# References


[1] T. Micro, “Assessing video game console security as the next generation approaches,” 22 November 2013. [Online]. Available: http://blog.trendmicro.com/assessing-video-game-console-security-next-generation-approaches/. [Accessed 27 April 2016].

[2] Wololo, “Linux on PS4: Fail0verflow showcase Linux on the PS4, run a Pokémon Demo (video),” wololo.net, 30 December 2015. [Online]. Available: http://wololo.net/2015/12/30/linux-on-the-ps4-fail0verflow-showcase-linux-running-on-the-ps4-run-a-pokemon-demo/. [Accessed 26 March 2016].

[3] Wololo, “Linux on PS4: Fail0verflow publish their fork of the linux Kernel (WIP),” wololo.net, 4 January 2016. [Online]. Available: http://wololo.net/2016/01/04/linux-on-ps4-fail0verflow-publish-their-fork-of-the-linux-kernel-wip/. [Accessed 26 March 2016].

[4] Wololo, “PS4 BadIRET Kernel exploit leaked,” Wololo.net, 2 March 2016. [Online]. Available: http://wololo.net/2016/03/02/ps4-badiret-kernel-exploit-leaked/. [Accessed 25 March 2016].

[5] Apple Inc, “WebKit,” WebKit, 2016. [Online]. Available: https://webkit.org/. [Accessed 24 April 2016].

[6] Wololo, “Xbox One Hack – Is Team Xecuter working on an Xbox One mod?,” wololo.net, 15 February 2016. [Online]. Available: http://wololo.net/2016/02/15/xbox-one-hack-is-team-xecuter-working-on-an-xbox-one-mod/. [Accessed 26 April 2016].

[7] Wololo, “Xbox One hack – Piracy trick from Indonesia promises up to 30 games for $65,” wololo, 5 February 2016. [Online]. Available: http://wololo.net/2016/02/05/xbox-one-hack-piracy-trick-originates-in-indonesia-promises-up-to-30-games-for-65/. [Accessed 26 April 2016].

[8] “PlayStation®4 (PS4™) Sells Through 5.7 Million Units Worldwide During The 2015 Holiday Season,” PR Newswire, 5 January 2016. [Online]. Available: http://www.prnewswire.com/news-releases/playstation4-ps4-sells-through-57-million-units-worldwide-during-the-2015-holiday-season-300199273.html. [Accessed 27 March 2016].

[9] V. McKalin, “Microsoft Sold 20 Million Xbox One Units To Consumers?,” Tech Times, 1 February 2016. [Online]. Available: http://www.techtimes.com/articles/128204/20160201/microsoft-sold-20-million-xbox-one-units-to-consumers.htm. [Accessed 27 March 2016].

[10] Nintendo, “Hardware and Software Sales Units,” 2016. [Online]. Available: https://www.nintendo.co.jp/ir/en/sales/hard_soft/index.html. [Accessed 23 April 2016].

[11] W. W. Ridgewell, *Determination and Exploitation of Potential Security Vulnerabilities in Networked Game Devices*, Athabasca, Alberta, 2011.

[12] H. Myrmo, *Game Consoles - Are they secure?,* Gjøvik, 2007.

[13] B. Gilbert, "Nintendo Wii U arrives in the US on Nov. 18 in two versions for $300 and $350, Europe on Nov. 30," Engadget, 9 September 2012. [Online]. Available: http://www.engadget.com/2012/09/13/wii-u-price-date-us/. [Accessed 27 March 2016].

[14] E. Johnson, "PlayStation 4 Will Be Released on November 15, Xbox One Release Date Still MIA," All Things D, 20 August 2013. [Online]. Available: http://allthingsd.com/20130820/playstation-4-will-be-released-on-november-15-xbox-one-release-date-still-mia/. [Accessed 27 March 2016].

[15] BBC, "Xbox One console release date set by Microsoft," 4 September 2013. [Online]. Available: http://www.bbc.com/news/technology-23961357. [Accessed 27 March 2016].

[16] C. Simpson, "AMD opens up about the PlayStation 4's custom processor," PC World, 22 February 2013. [Online]. Available: http://www.pcworld.idg.com.au/article/454508/amd_opens_up_about_playstation_4_custom_processor/. [Accessed 27 March 2016].

[17] A. L. Shimpi, "The Xbox One: Hardware Analysis & Comparison to PlayStation 4," Anand Tech, 22 May 2013. [Online]. Available: http://www.anandtech.com/show/6972/xbox-one-hardware-compared-to-playstation-4/2. [Accessed 27 March 2016].

[18] Sony Computer Entertainment, "PlayStation 4 Specifications," 27 March 2016. [Online]. Available: https://www.playstation.com/en-us/explore/ps4/techspecs/. [Accessed 27 March 2016].

[19] Sony Interactive Entertainment, "PS4 Remote Play and Second Screen," 2016. [Online]. Available: https://support.us.playstation.com/articles/en_US/KC_Article/PS4-Remote-Play-and-Second-Screen/?. [Accessed 25 April 2016].

[20] Sony Interactive Entertainment, "PS4: Remote Play for PC and Mac," 2016. [Online]. Available: https://support.us.playstation.com/articles/en_US/KC_Article/PS4-Remote-Play-for-PC-and-Mac. [Accessed 25 April 2016].

[21] S. Andrews, "PS4 vs Xbox One review: Which is best - PlayStation 4 or Xbox One? PlayStation update brings PC/Mac Remote Play along with a number of improvements," Tech Advisor, 17 March 2016. [Online]. Available: http://www.pcadvisor.co.uk/review/games-consoles/ps4-vs-xbox-one-review-2016-best-cross-platform-play-3452022/. [Accessed 27 March 2016].

[22] "Xbox One Hardware Specs," IGN Entertainment Games, July 2015. [Online]. Available: http://uk.ign.com/wikis/xbox-one/Xbox_One_Hardware_Specs. [Accessed 27 March 2016].

[23] Microsoft, "How to use game streaming in the Xbox app on Windows 10," Microsoft, 2016. [Online]. Available: https://support.xbox.com/en-US/xbox-on-windows/gaming-on-windows/how-to-use-game-streaming. [Accessed 25 April 2016].

[24] Microsoft, "Set up and use SmartGlass on Xbox One," Microsoft, 2016. [Online]. Available: https://support.xbox.com/en-US/xbox-one/apps/smartglass-console-setup. [Accessed 25 April 2016].

[25] J. Hruska, “Nintendo tears down Wii U to show off single-chip IBM/AMD CPU + GPU,” Ziff Davis, LLC, 12 October 2012. [Online]. Available: http://www.extremetech.com/gaming/137746-nintendo-tears-down-wii-u-to-show-off-single-chip-ibmamd-cpu-gpu. [Accessed 24 April 2016].

[26] P. Hernandez, “Wii U to Have 2GB Internal Memory, GPGPU Support,” NintendoWorldReport, 13 September 2012. [Online]. Available: http://www.nintendoworldreport.com/news/31665/wii-u-to-have-2gb-internal-memory-gpgpu-support. [Accessed 24 April 2016].

[27] Nintendo, “Wii U technical specs,” Nintendo, 2016. [Online]. Available: https://www.nintendo.com/wiiu/features/tech-specs/. [Accessed 24 April 2016].

[28] H. R. K. X. I. S. M. Davies, “Forensic analysis of a Sony PlayStation 4: A first look,” *Digital Investigation,* vol. 12, no. 1, pp. 81-89, 2015.

[29] I. B. A. M. A. R. J. Moore, “Preliminary forensic analysis of the Xbox One,” *Digital Investigation,* vol. 11, no. 2, pp. 57-65, 2014.

[30] Microsoft, “Microsoft Security Development Lifecycle (SDL) – Process Guidance,” Microsoft, 2012. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/84aed186-1d75-4366-8e61-8d258746bopq.aspx. [Accessed 29 March 2016].

[31] Microsoft, “SDL Process: Design,” Microsoft, 2016. [Online]. Available: https://www.microsoft.com/en-us/sdl/process/design.aspx. [Accessed 29 March 2016].

[32] Microsoft, “Introduction to Threat Modeling,” 2016. [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=16420. [Accessed 29 March 2016].

[33] Microsoft, “SDL Threat Modeling Tool,” Microsoft, 2016. [Online]. Available: https://www.microsoft.com/en-us/sdl/adopt/threatmodeling.aspx. [Accessed 30 March 2016].

[34] MyAppSecurity, “Comparison of Threat Modeling Methodologies,” 28 May 2012. [Online]. Available: http://myappsecurity.com/comparison-threat-modeling-methodologies/. [Accessed 30 March 2016].

[35] T. UcedaVelez, “Real World Threat Modeling Using the PASTA Methodology,” 2012. [Online]. Available: https://www.owasp.org/images/a/aa/AppSecEU2012_PASTA.pdf. [Accessed 30 March 2016].

[36] E. S. S. S. Brenda Larcom, “Trike,” [Online]. Available: http://octotrike.org/. [Accessed 30 March 2016].

[37] B. L. M. E. Paul Saitta, “Trike v.1 Methodology Document [Draft],” 13 July 2005. [Online]. Available: http://octotrike.org/papers/Trike_v1_Methodology_Document-draft.pdf. [Accessed 30 March 2016].

[38] E. S. Brenda Larcom, “Trike,” 22 April 2013. [Online]. Available: https://sourceforge.net/projects/trike/. [Accessed 30 March 2016].

[39] OWASP, “Application Threat Modeling,” OWASP, 8 March 2015. [Online]. Available: https://www.owasp.org/index.php/Application_Threat_Modeling. [Accessed 15 April 2016].

[40] A. Shostack, Threat Modeling, Indianapolis, Indiana: John Wiley & Sons, Inc., 2014.

[41] SmartDraw, LLC, “Data Flow Diagram,” SmartDraw, 2016. [Online]. Available: https://www.smartdraw.com/data-flow-diagram/. [Accessed 24 April 2016].

[42] Microsoft Corporation, “The STRIDE Threat Model,” Microsoft, 2005. [Online]. Available: https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx. [Accessed 23 April 2016].

[43] OWASP, “STRIDE,” 8 March 2015. [Online]. Available: https://www.owasp.org/index.php/Application_Threat_Modeling#STRIDE. [Accessed 23 April 2016].

[44] OWASP, “Threat Analysis,” OWASP, 8 March 2015. [Online]. Available: https://www.owasp.org/index.php/Application_Threat_Modeling#Threat_Analysis. [Accessed 23 April 2016].

[45] Microsoft, “Step 6. Rate the Threats,” June 2003. [Online]. Available: https://msdn.microsoft.com/en-us/library/aa302419.aspx#c03618429_011. [Accessed 23 April 2016].

[46] OWASP, “OWASP Risk Rating Methodology,” OWASP, 3 September 2015. [Online]. Available: https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology. [Accessed 23 April 2016].

[47] FIRST.org Inc., “Common Vulnerability Scoring System, V3 Development Update,” 2016. [Online]. Available: http://www.first.org/cvss. [Accessed 22 April 2016].

[48] The MITRE Corporation, “Common Weakness Scoring System (CWSS™),” 5 September 2014. [Online]. Available: https://cwe.mitre.org/cwss/cwss_v1.0.1.html. [Accessed 23 April 2016].

[49] Microsoft, “Microsoft Threat Modeling Tool 2016,” Microsoft, 6 October 2015. [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=49168. [Accessed 21 April 2016].

[50] Microsoft, “Threat Modeling Tool 2016 Getting Started Guide,” 6 October 2015. [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=49168. [Accessed 21 April 2016].

[51] OWASP, “OWASP Risk Rating Template Example,” 1 October 2015. [Online]. Available: https://www.owasp.org/index.php/File:OWASP_Risk_Rating_Template_Example.xlsx. [Accessed 22 April 2016].

[52] R. Matulevičius, “Fundamentals of Secure System Modelling,” *Chapters 7 and 8 (unpublished draft),* pp. 105-116, 2016.

[53] Gliffy, “Gliffy,” Gliffy, Inc, 2016. [Online]. Available: https://www.gliffy.com/. [Accessed 22 April 2016].

[54] Sony Interactive Entertainment Inc., “PlayStation®App Connection Settings,” 2016. [Online]. Available: http://manuals.playstation.net/document/en/ps4/settings/app.html. [Accessed 22 April 2016].

[55] Sony Interactive Entertainment Inc., “Connecting from other devices,” 2016. [Online]. Available: http://manuals.playstation.net/document/en/ps4/basic/remoteplay.html. [Accessed 22 April 2016].

[56] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

# Appendix 1 – The HGCTM implementation guide

## Implementation of the framework

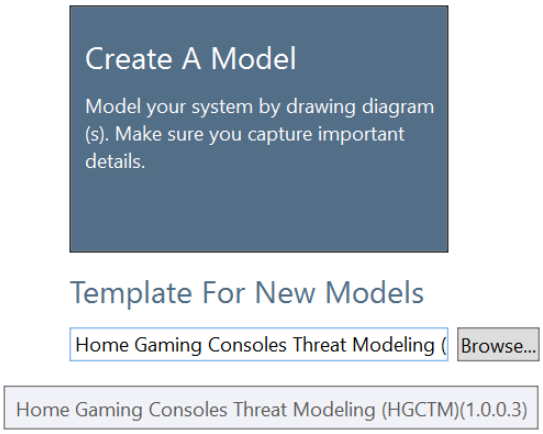
For successful implementation, following tools, templates and documentations are used:

- The Microsoft Threat Modeling Tool (TMT) 2016 with documentation;
- Home Gaming Consoles Threat Modeling (HGCTM) TMT 2016 template;
- Excel template for OWASP Risk Rating Methodology;
- Misuse cases modeling tool depending on the needs or availability:
    - Any modeling tool supporting Unified Modeling Language;
    - Online Diagram and Flowchart Software called Gliffy.;
- Misuse cases modeling guide and syntax from "Fundamentals of Secure System Modelling" unpublished draft by R. Matulevičius.

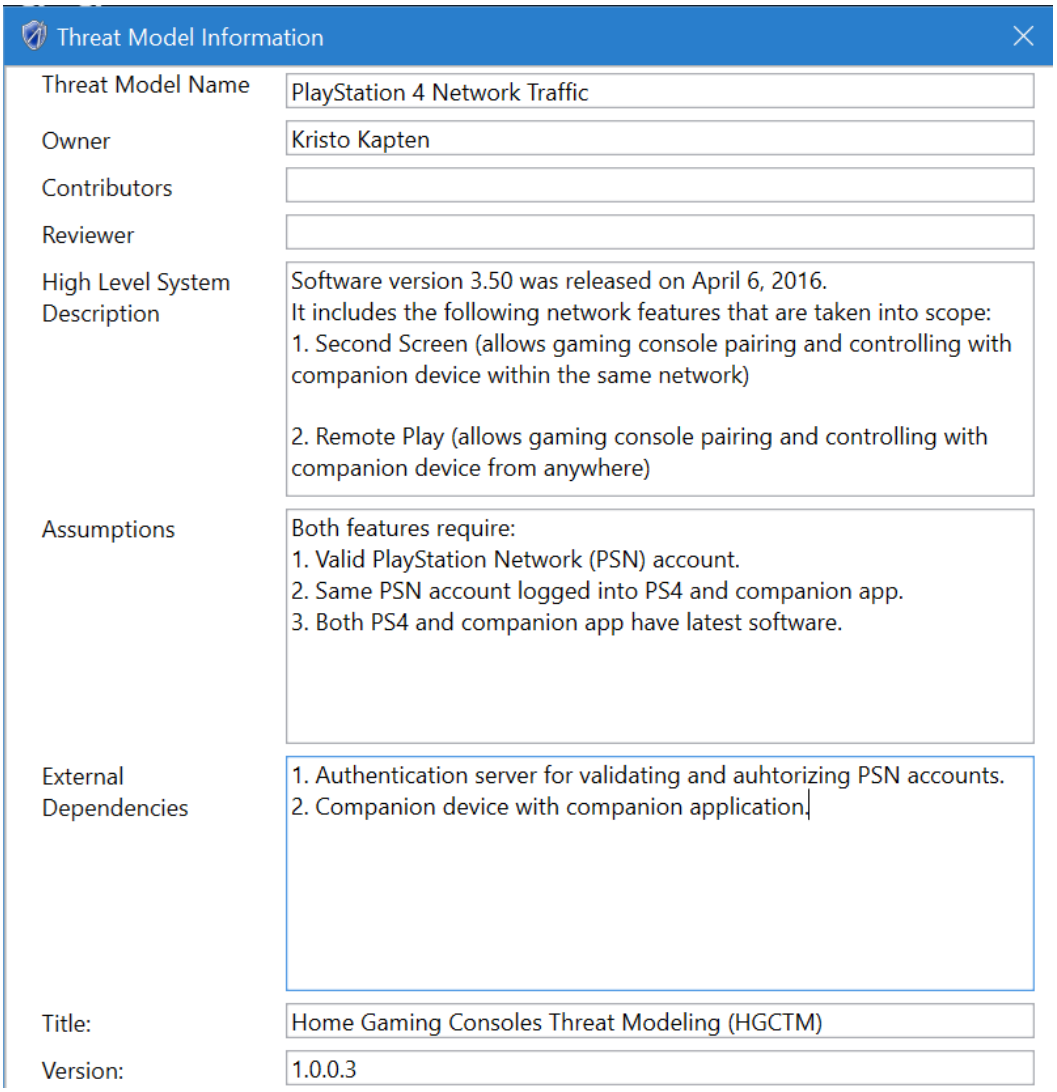
## 1. Model the system

1. Open the Threat Modeling Tool 2016 and choose the HGCTM template for new models.

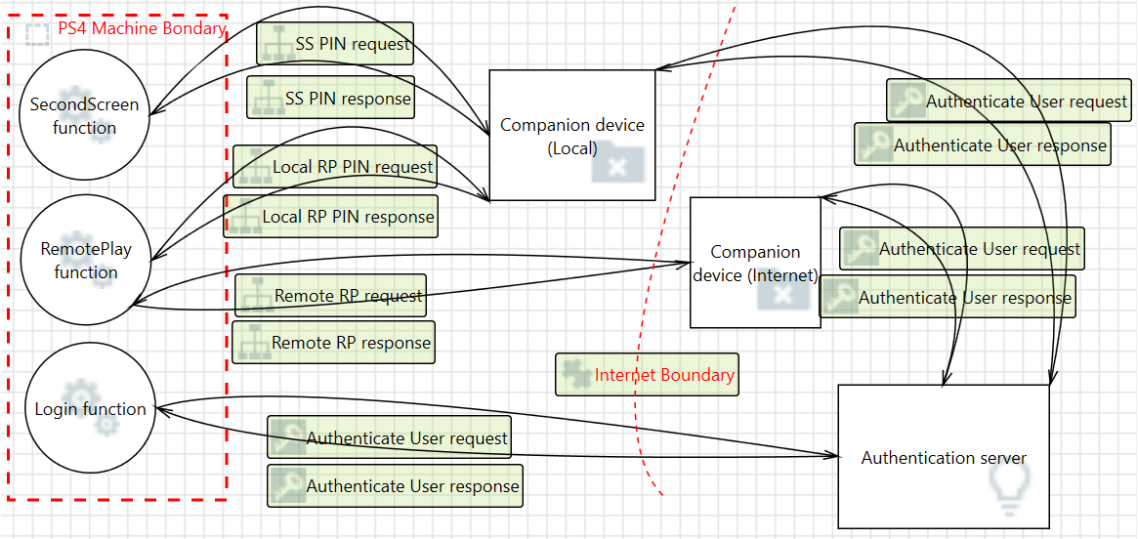
2. Choose the “Create A Model.”




3. Choose “File” and “Threat Model Information” within the Threat Modeling Tool 2016 and fill in the Scope, Assumptions and External Dependencies details like in the sample below.



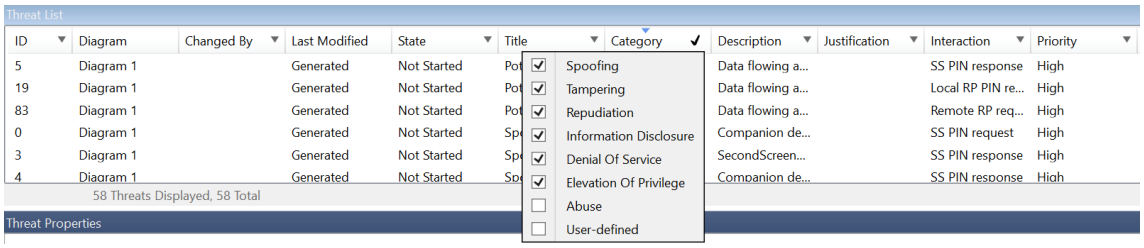

4. Close the Threat Model information window and start modeling the Data Flow Diagrams with the help from Threat Modeling Tool's user guide. Outcome should be similar to the sample below.



## 2. Determine the threats

1. Switch to “Analysis View” and it provides the list of automatically generated threats

2. Filter threats according to STRIDE, like the sample below.

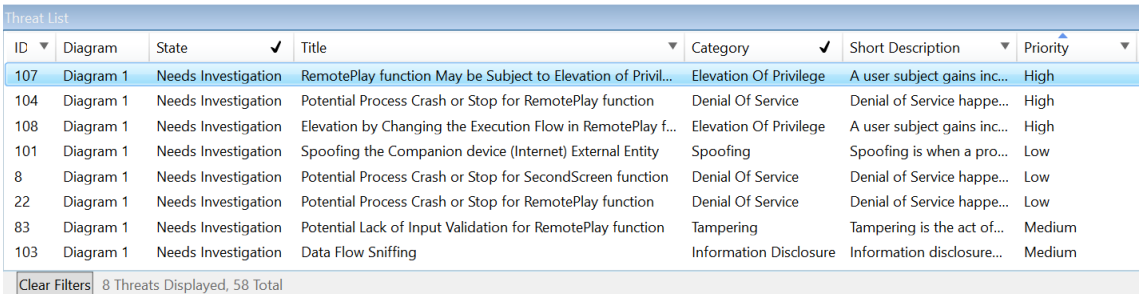



## 3. Address the threats

1. Choose state for each threat in the list either “Needs Investigating”, “Not Applicable” or “Mitigated.” The not applicable ones can be removed from the list or kept for safe keeping for any future updates to the DFD or template.

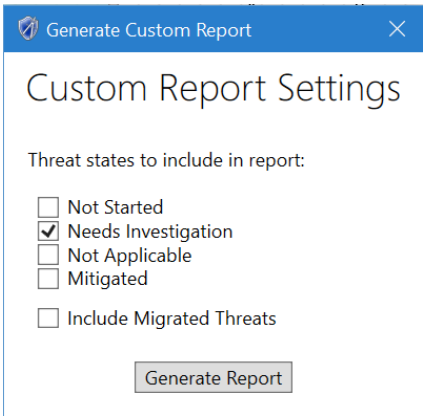
2. Filter threat according to “Needs Investigating.”

3. Use OWASP Risk Rating Methodology’s excel and set priority level for each threat either low, medium or high. An example of filtered and rated threats below.




4. Choose “Reports” and “Generate Custom Report.”

5. Select “Need investigation” and “Generate Report.”




## 4. Modeling the test cases

1. Select highest priority threat or the most interesting one from the Custom Report.

2. Fill in the Misuse Case descriptive table for each Potential Threat using the tables below.

| **Misuse Case item** | **Potential Threat** |
|---|---|
| System boundary | |
| Use case | |
| Vulnerability | |
| Misuse case | |
| Impact | |
| Security criterion | |

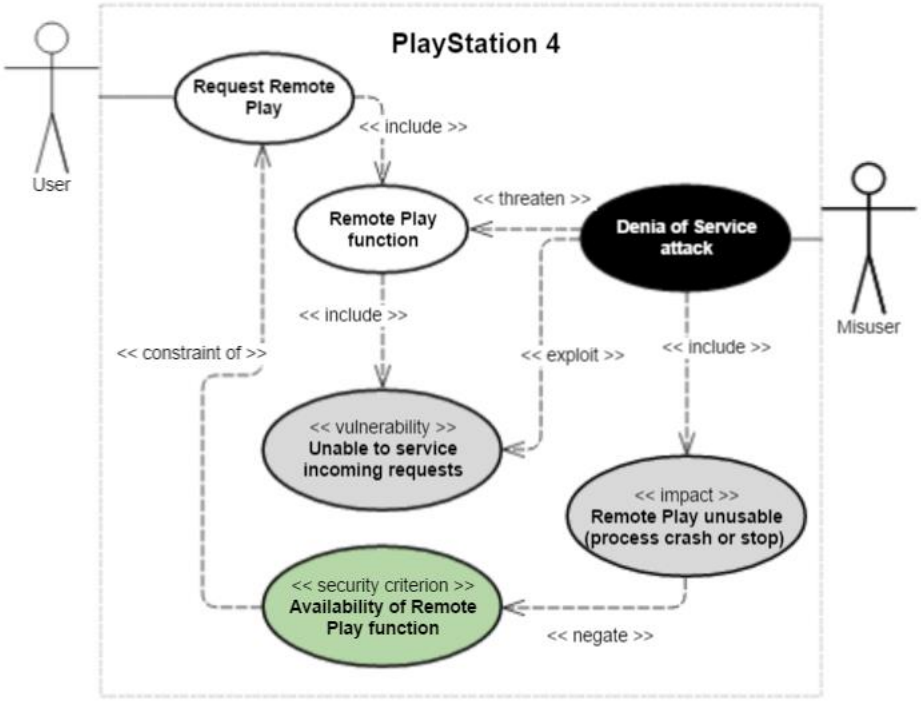
3. Identify security criterion for each threat according to STRIDE threat type in the Custom Report and the table below.

| **Threat type (STRIDE)** | **Security Criterion** |
|---|---|
| Spoofing | Authentication |
| Tampering | Integrity |
| Repudiation | Non-repudiation |
| Information Disclosure | Confidentiality |
| Denial of Service | Availability |
| Elevation of Privilege | Authorization |

4. Identify all other Misuse Case items for each threat according to Custom Report and the following table below.

| **Misuse Case item** | **According to Potential Threats** |
|---|---|
| System boundary | Gaming Console<br>(PlayStation 4, Xbox One, Wii U) |
| Use case | Action how valid user interacts with the Gaming Console or its feature. |
| Vulnerability | Weakness or a flaw in Gaming Console's security. |
| Misuse case | Attack method. |
| Impact | The potential negative consequence. |
| Security criterion | Taken from the Table 2: STRIDE Threat type to Security Criterion |

5. Model Misuse cases for each chosen threat according to descriptive table just filled and using Unified Modeling Language (UML) modeling tool with Misuse cases syntax. Misuse case should be similar to the sample below.




6. Manually move all created Misuse cases descriptive tables and models into the Custom Report previously generated.

## Appendix 2 – PlayStation 4 Network Traffic HGCTM report

# Threat Modeling Report

Created on 23.04.2016

Threat Model Name: PlayStation 4 Network Traffic

Owner: Kristo Kapten

Reviewer:

Contributors:

Description: Software version 3.50 was released on April 6, 2016. It includes the following network features that are taken into scope: 1. Second Screen (allows gaming console pairing and controlling with companion device within the same network) 2. Remote Play (allows gaming console pairing and controlling with companion device from anywhere)

Assumptions: Both features require: 1. Valid PlayStation Network (PSN) account. 2. Same PSN account logged into PS4 and companion app. 3. Both PS4 and companion app have latest software.

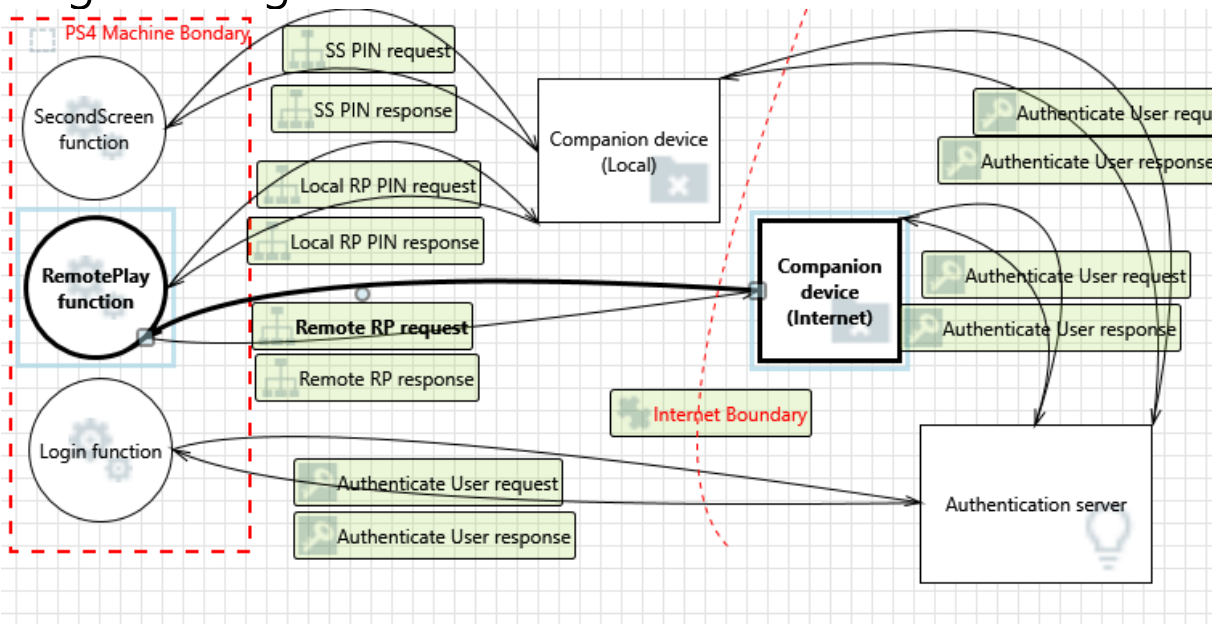
External Dependencies: 1. Authentication server for validating and authorizing PSN accounts. 2. Companion device with companion application.

### Threat Model Summary:

| | |
|---|---|
| Not Started | 0 |
| Not Applicable | 0 |
| Needs Investigation | 8 |
| Mitigation Implemented | 0 |
| Total | 8 |

---

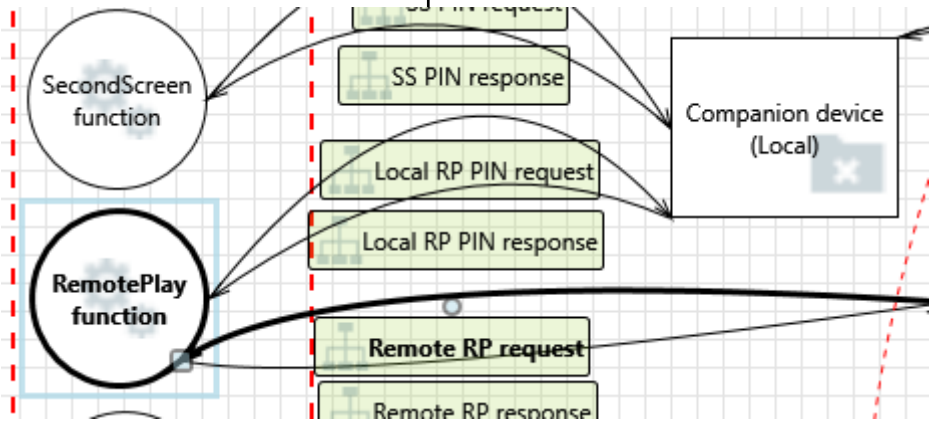
# Diagram: Diagram 1




## Diagram 1 Diagram Summary:

| | |
|---|---|
| Not Started | 0 |
| Not Applicable | 0 |
| Needs Investigation | 8 |
| Mitigation Implemented | 0 |
| Total | 8 |

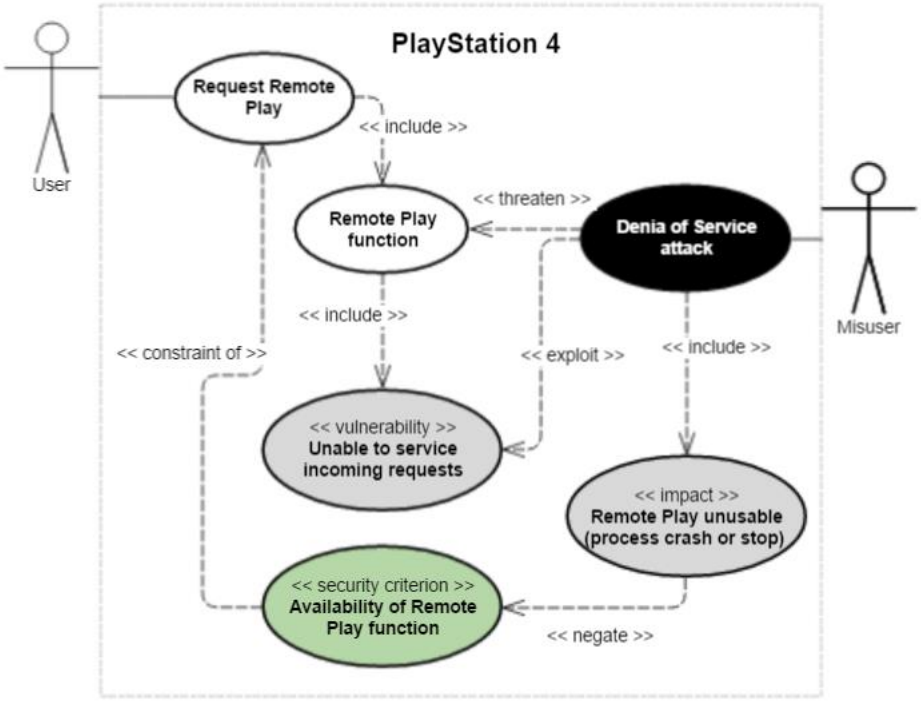
## Interaction: Local RP PIN response



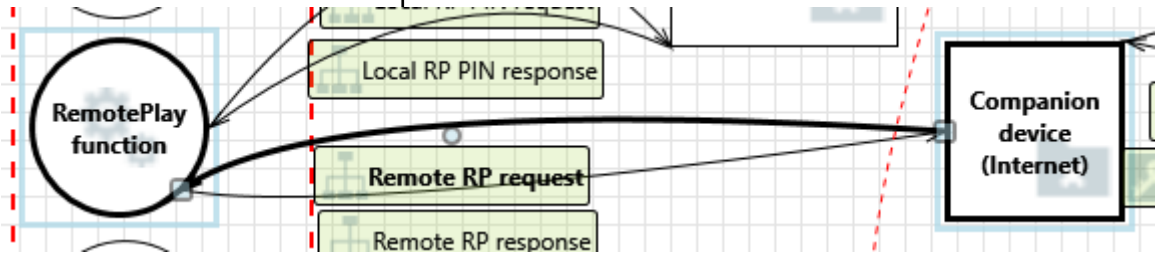
1. Potential Process Crash or Stop for RemotePlay function [State: Needs Investigation] [Priority: Low]

| | |
|---|---|
| Category: | Denial Of Service |
| Description: | RemotePlay function crashes, halts, stops or runs slowly; in all cases violating an availability metric. |
| Justification: | <no mitigation provided> |
| Short Description: | Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec. |

| **Misuse Case item** | **Potential Threat** |
|---|---|
| System boundary | PlayStation 4 |
| Use case | User request Remote Play with companion device. |
| Vulnerability | Process or a datastore is not able to service incoming requests or perform up to spec. |
| Misuse case | Denial of Service against Remote Play function. |
| Impact | Potential Process Crash or Stop for Remote Play function. |
| Security criterion | Availability |



Interaction: Remote RP request




2. Potential Lack of Input Validation for RemotePlay function [State: Needs Investigation] [Priority: Medium]

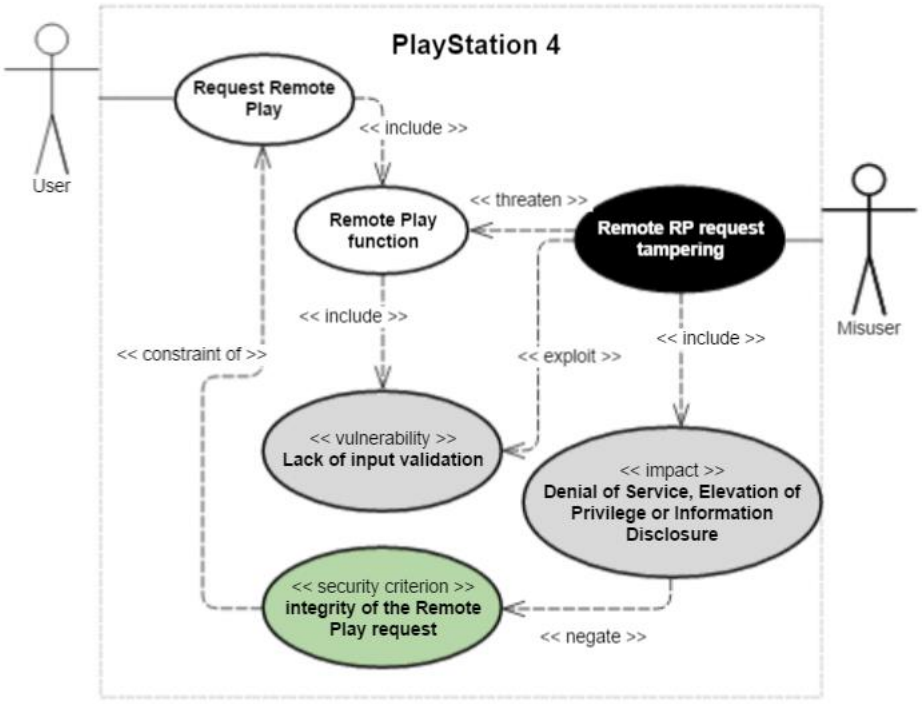
Category: Tampering

Description: Data flowing across Remote RP request may be tampered with by an attacker. This may lead to a denial of service attack against RemotePlay function or an elevation of privilege attack against RemotePlay function or an information disclosure by RemotePlay function. Failure to verify that input is as expected is a root cause of a very large number of exploitable issues. Consider all paths and the way they handle data. Verify that all input is verified for correctness using an approved list input validation approach.

Justification: <no mitigation provided>

Short Description: Tampering is the act of altering the bits. Tampering with a process involves changing bits in the running process. Similarly, Tampering with a data flow involves changing bits on the wire or between two running processes.

| **Misuse Case item** | **Potential Threat** |
|---|---|
| System boundary | PlayStation 4 |
| Use case | User request Remote Play with companion device (Internet). |
| Vulnerability | Potential Lack of Input Validation for RemotePlay function. |
| Misuse case | Remote RP request may be tampered with by an attacker. |
| Impact | May lead to denial of service attack against RemotePlay function or an elevation of privilege attack against RemotePlay function or an information disclosure by RemotePlay function. |
| Security criterion | Integrity |




3. Elevation by Changing the Execution Flow in RemotePlay function [State: Needs Investigation] [Priority: High]

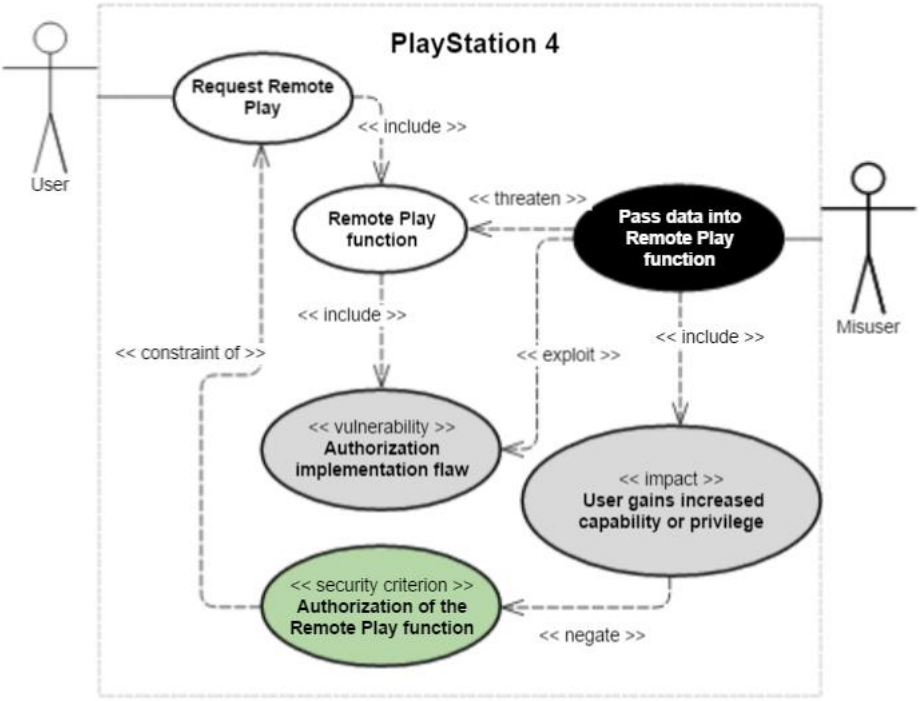
Category: Elevation Of Privilege

Description: An attacker may pass data into RemotePlay function in order to change the flow of program execution within RemotePlay function to the attacker's choosing.

Justification: <no mitigation provided>

Short Description: A user subject gains increased capability or privilege by taking advantage of an implementation bug.

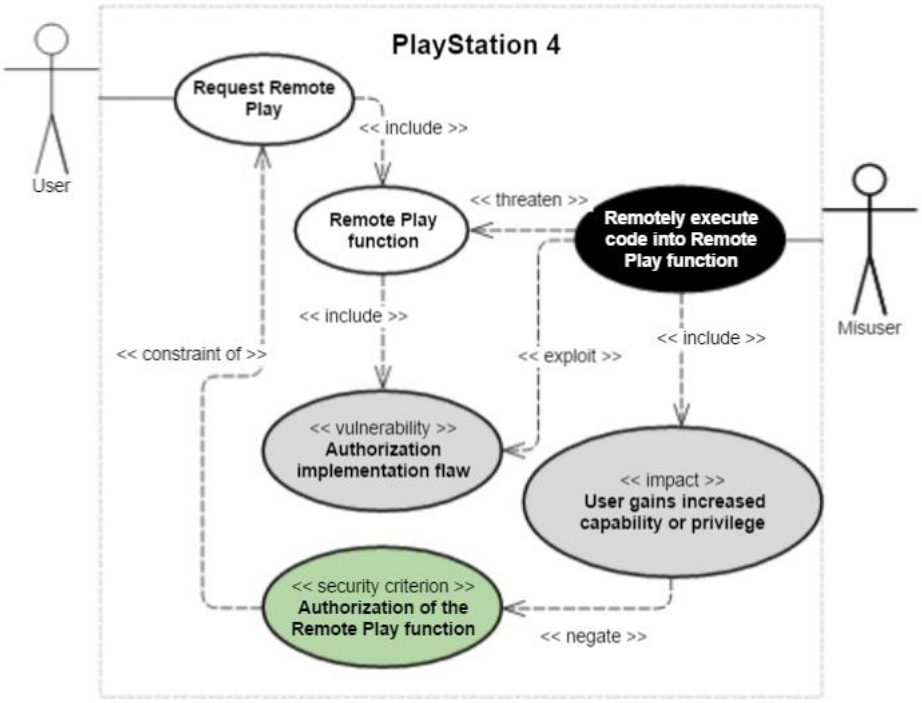
| **Misuse Case item** | **Potential Threat** |
|---|---|
| System boundary | PlayStation 4 |
| Use case | User request Remote Play with companion device (Internet). |
| Vulnerability | Authorization implementation flaw in RemotePlay function. |
| Misuse case | Attacker passes data into RemotePlay function in order to change the flow of program execution within RemotePlay function to the attacker's choosing. |
| Impact | User subject gains increased capability or privilege. |
| Security criterion | Authorization. |




4. RemotePlay function May be Subject to Elevation of Privilege Using Remote Code Execution [State: Needs Investigation] [Priority: High]

| | |
|---|---|
| Category: | Elevation Of Privilege |
| Description: | Companion device (Internet) may be able to remotely execute code for RemotePlay function. |
| Justification: | <no mitigation provided> |
| Short Description: | A user subject gains increased capability or privilege by taking advantage of an implementation bug. |

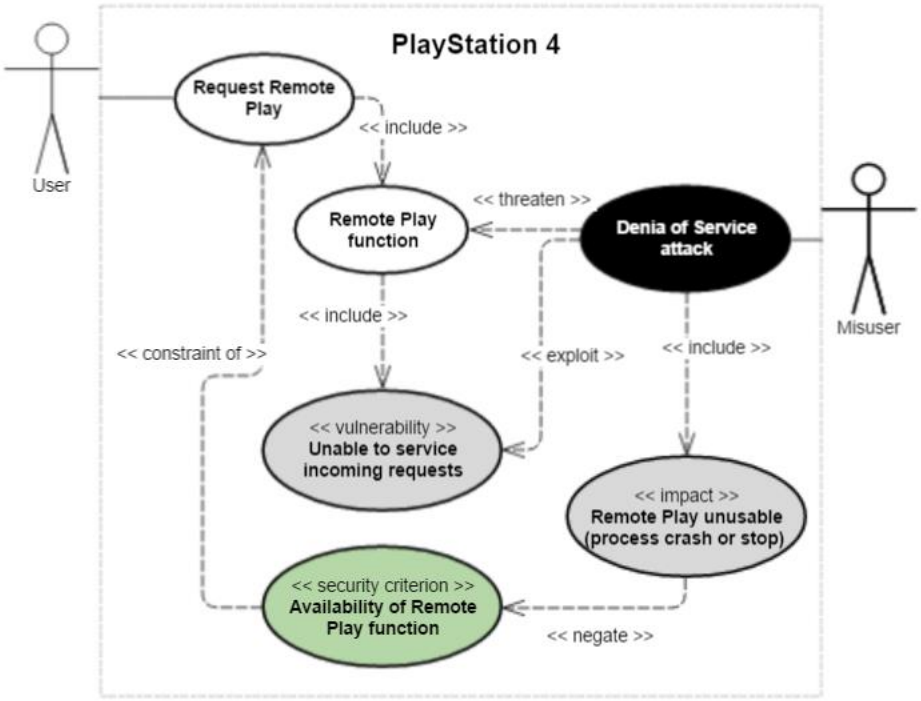
| **Misuse Case item** | **Potential Threat** |
|---|---|
| System boundary | PlayStation 4 |
| Use case | User request Remote Play with companion device (Internet). |
| Vulnerability | Authorization implementation flaw in RemotePlay function. |
| Misuse case | Attacker may be able to remotely execute code for RemotePlay function. |
| Impact | User subject gains increased capability or privilege. |
| Security criterion | Authorization. |




5. Potential Process Crash or Stop for RemotePlay function [State: Needs Investigation] [Priority: High]

| | |
|---|---|
| Category: | Denial Of Service |
| Description: | RemotePlay function crashes, halts, stops or runs slowly; in all cases violating an availability metric. |
| Justification: | <no mitigation provided> |
| Short Description: | Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec. |

| **Misuse Case item** | **Potential Threat** |
|---|---|
| System boundary | PlayStation 4 |
| Use case | User request Remote Play with companion device (Internet). |
| Vulnerability | Process or a datastore is not able to service incoming requests or perform up to spec. |
| Misuse case | Denial of Service against Remote Play function. |
| Impact | Potential Process Crash or Stop for Remote Play function. |
| Security criterion | Availability |




6. Data Flow Sniffing [State: Needs Investigation] [Priority: Medium]

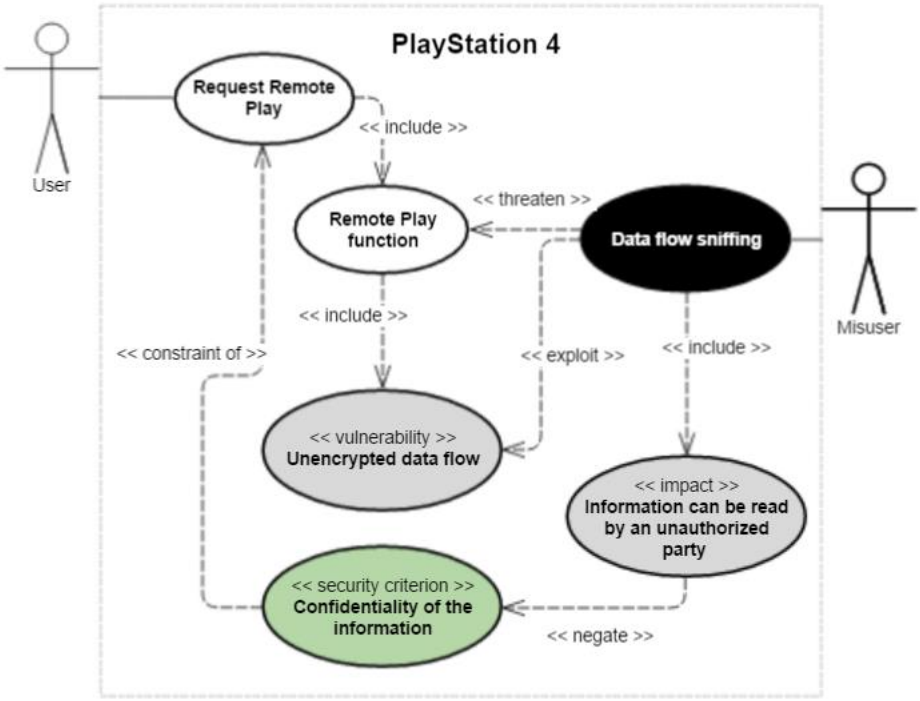
Category: Information Disclosure

Description: Data flowing across Remote RP request may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.

Justification: <no mitigation provided>

Short Description: Information disclosure happens when the information can be read by an unauthorized party.

| **Misuse Case item** | **Potential Threat** |
|---|---|
| System boundary | PlayStation 4 |
| Use case | User request Remote Play with companion device (Internet). |
| Vulnerability | Unencrypted data flow. |
| Misuse case | Data flowing across Remote RP request may be sniffed by an attacker. |
| Impact | Information can be read by an unauthorized party. |
| Security criterion | Confidentiality |




7. Spoofing the Companion device (Internet) External Entity [State: Needs Investigation] [Priority: Low]

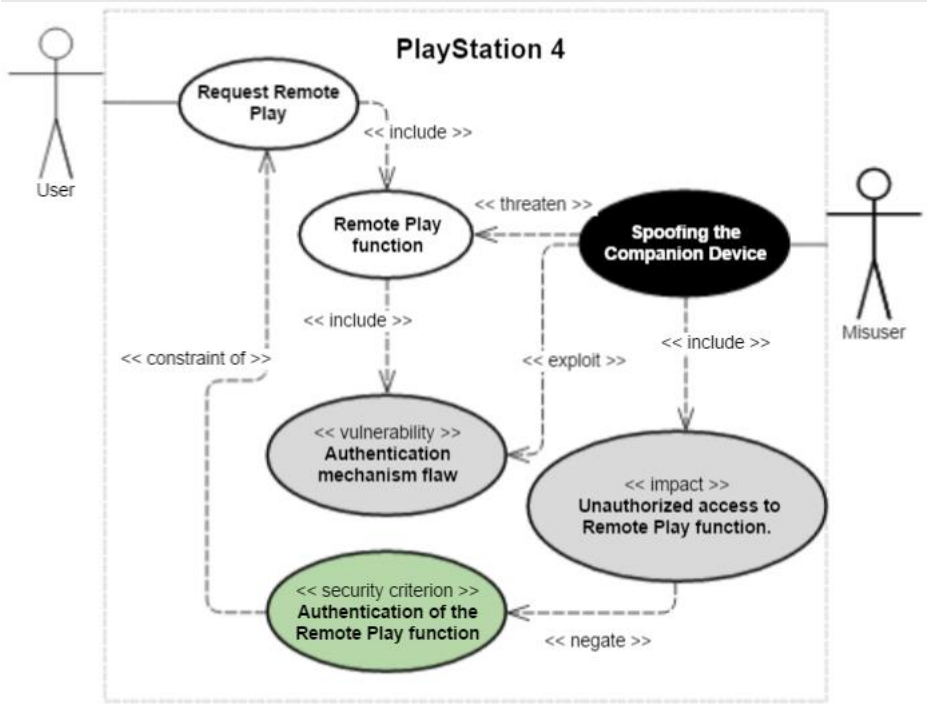
Category: Spoofing

Description: Companion device (Internet) may be spoofed by an attacker and this may lead to unauthorized access to RemotePlay function. Consider using a standard authentication mechanism to identify the external entity.

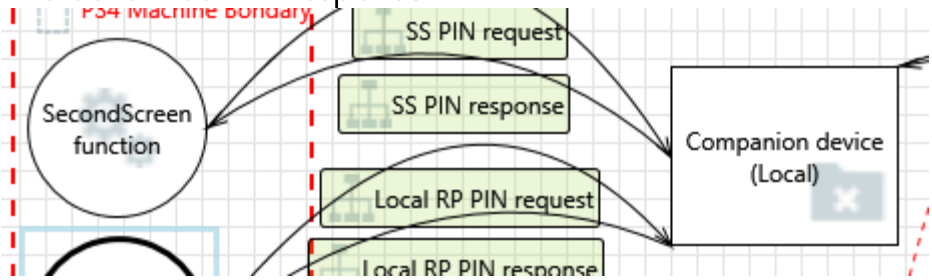
Justification: <no mitigation provided>

Short Description: Spoofing is when a process or entity is something other than its claimed identity. Examples include substituting a process, a file, website or a network address.

| **Misuse Case item** | **Potential Threat** |
|---|---|
| System boundary | PlayStation 4 |
| Use case | User request Remote Play with companion device (Internet). |
| Vulnerability | Flaw in authentication mechanism that identifies the external entity. |
| Misuse case | Companion device (Internet) may be spoofed by an attacker. |
| Impact | Unauthorized access to RemotePlay function. |
| Security criterion | Authentication |



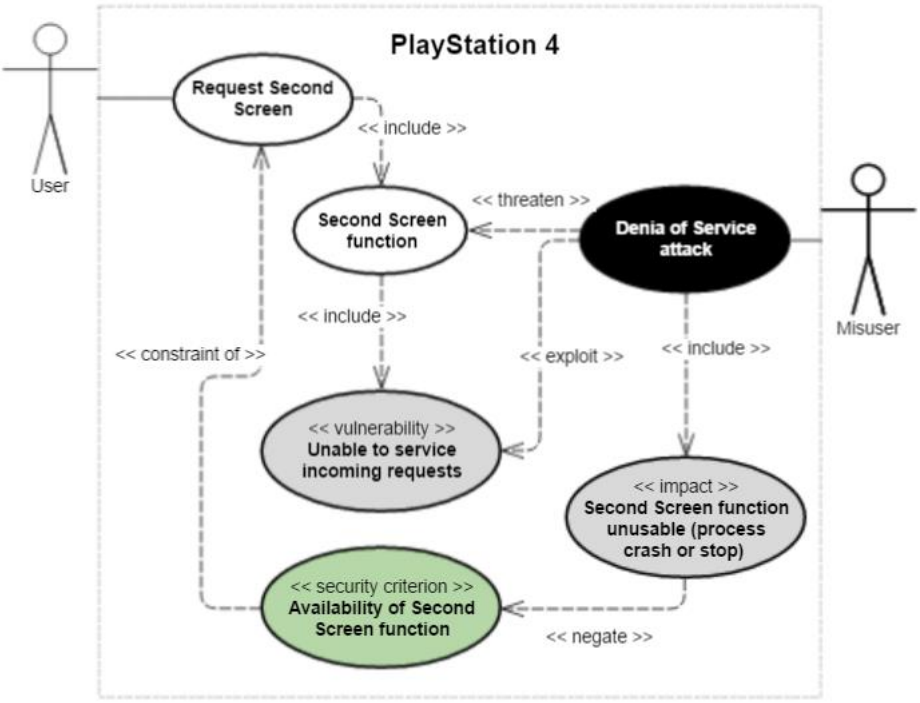

Interaction: SS PIN response




8. Potential Process Crash or Stop for SecondScreen function [State: Needs Investigation] [Priority: Low]

| | |
|---|---|
| Category: | Denial Of Service |
| Description: | SecondScreen function crashes, halts, stops or runs slowly; in all cases violating an availability metric. |
| Justification: | <no mitigation provided> |
| Short Description: | Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec. |

| Misuse Case item | Potential Threat |
|---|---|
| System boundary | PlayStation 4 |
| Use case | User request Second Screen with companion device. |
| Vulnerability | Process or a datastore is not able to service incoming requests or perform up to spec. |
| Misuse case | Denial of Service against Second Screen function. |
| Impact | Potential Process Crash or Stop for Second Screen function. |
| Security criterion | Availability |


PlayStation 4
User
Request Second Screen
<< include >>
Second Screen function
<< threaten >>
Denia of Service attack
Misuser
<< include >>
<< constraint of >>
<< exploit >>
<< include >>
<< vulnerability >>
Unable to service incoming requests
<< impact >>
Second Screen function unusable (process crash or stop)
<< security criterion >>
Availability of Second Screen function
<< negate >>