

RAMSES

RAMSES

Internet Forensic platform for tracking the money flow of financially-motivated malware

H2020 - 700326

D4.4 Optimal Model System Documentation

Lead Authors: Darren Hurley-Smith, Julio Hernandez-Castro, Edward Cartwright, and Anna Stepanova

Deliverable nature:	Demonstrator (D)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	28/02/2018
Actual delivery date:	27/02/2018
Version:	1.0
Total number of pages:	33
Keywords:	Economics, Ransomware, Model System, Python, Profit-Motive

Abstract

This report is a compilation of user documentation, functional specifications, and the mathematical models at the heart of deliverable 4.4 – and optimal economic model system for ransomware.

This report is delivered alongside a web GUI locally hosted application, providing the model system. This system, along with the documentation provided in this report and supplementary tutorial videos, is intended to provide LEA and scholarly users with a means of estimating the optimal profit of user defined or historical malware, in the context of economic theory.

[End of abstract]

Executive summary

The deliverable (4.4) is comprised of two elements: an optimal economic model tool, and documentation to support that tool.

The tool has been named the Optimal Economic Model System for Ransomware (OEMSR). It is currently in a version 1 state, with full functionality that meets the objectives of this deliverable. This tool provides users with the ability to define the abstract economic terms of their own malware, such as price, target selection criteria and ransom strategy. The model will then provide an evaluation of their chosen criteria, including distance from the optimal state for the given target population. It will also plot this optimal model for the given population and strategy selection, allowing users to visually compare their selection to the potential optimal. Example malware, based on real world strains, will be provided to allow evaluation of their optimality for a user defined population. Comparison of examples with user define malware is also supported. Users can also create custom populations driven by their own survey data, using the survey previously submitted in D4.1.

The target audience of this tool includes LEAs and scholars with an interest in Economic and/or Cyber-crime. We also anticipate that this tool will see further expansion, as there are opportunities to expand the economic model to work alongside epidemiological and psychological models. In its current state, this tool will enable analysts to more accurately predict the threat and profitability of hypothetical and real malware strains, leading to a better general state of awareness and readiness as cybercriminals attempt to innovate towards the optimal state.

This document is the deliverable report, providing full details on the usage, functional specifications and project-specific considerations. Usage is defined in the manner of a user manual, providing installation, access and interaction instructions. This part of the document also includes a general overview of critical systems, but stops short of functional or mathematical topics, which are not the province of this part of the report.

The functional specification provides a module-by-module technical breakdown of the system. This includes an overview of each class or function, their input arguments and the parameters that they return. The relationship between classes and functions is also discussed. This allows individuals with appropriate programming skills to identify potential opportunities for extension or revision of the tool, a process we expect to continue beyond the submission date. We point out several code refactoring opportunities, for this reason.

A general report finishes this document. This provides the reader with a breakdown of objectives, how they were achieved, measurable impacts, and the mathematical model at the heart of the OEMSR tool. This section of the document is intended to demonstrate the tool's compliance with the work plan and deliverable brief and highlight any additional value that has been incorporated into the work.

The objective of this deliverable, has been to develop an optimal economic model tool. This tool should be able to provide users with the ability to determine the distance from the optimal state for a given malware strain. They should also be able to compare their hypothetical malware with real world examples. These objectives have been met, and in some areas exceeded.

Document Information

IST Project Number	700326	Acronym	RAMSES
Full Title	Internet Forensic platform for tracking the money flow of financially-motivated malware		
Project URL	http://www.ramses2020.eu		
EU Project Officer	Nada Milisavljevic		

Deliverable	Number	D4.4	Title	Optimal Model System
Work Package	Number	WP4	Title	Modelling Ransomware from the point of view of Economic Theory and Applications

Date of Delivery	Contractual	M18	Actual	M18
Status	version x1.0		final <input type="checkbox"/>	
Nature	demonstrator <input type="checkbox"/>			
Dissemination level	public <input type="checkbox"/>			

Authors (Partner)	University of Kent			
Responsible Author	Name	Darren Hurley-Smith	E-mail	d.hurley-smith@kent.ac.uk
	Partner	University of Kent	Phone	07870806745

Abstract (for dissemination)	<p>This report is a compilation of user documentation, functional specifications, and the mathematical models at the heart of deliverable 4.4 – and optimal economic model system for ransomware.</p> <p>This report is delivered alongside a web GUI locally hosted application, providing the model system. This system, along with the documentation provided in this report and supplementary tutorial videos, is intended to provide LEA and scholarly users with a means of estimating the optimal profit of user defined or historical malware, in the context of economic theory.</p>
Keywords	Economics, Ransomware, Model System, Python, Profit-Motive

Version Log			
Issue Date	Rev. No.	Author	Change
01/12/2017	0.1	Darren Hurley-Smith	Initial draft system using dummy variables
15/01/2018	0.2	Darren Hurley-Smith	First test system using survey estimates
20/02/2018	0.3	Darren Hurley-Smith	Full system, system documentation and functional specification completed and proof-read
26/02/2018	1.0	Darren Hurley-Smith	Final system, including internal recommendations from Julio Hernandez-Castro and Edward Cartwright. Full documentation, videos, functional specification, and model overview completed.

Table of Contents

Executive summary.....	3
Document Information.....	4
Table of Contents.....	5
List of figures.....	7
1 Introduction.....	8
2 User Manual.....	9
2.1 Introduction to the OEMSR.....	9
2.1.1 Developers.....	9
2.2 Start-up, Dependencies and First-Run Guide.....	10
2.2.1 Directory Set-Up.....	10
2.2.2 Installing Dependencies.....	10
2.2.3 First Run.....	10
2.3 Using the OEMSR.....	12
2.3.1 Population Definition.....	13
2.3.2 Population Types.....	13
2.3.3 Type of Ransom.....	14
2.3.4 Type of Discrimination.....	14
2.3.5 Ransom, Brackets and Cost inputs.....	14
2.3.6 Example Malware Selection.....	15
2.3.7 Results.....	16
2.4 Summary.....	21
3 Functional Definition of the Optimal Model System.....	22
3.1 Model_front.py.....	22
3.1.1 Class: Reusableform(Form).....	22
3.1.2 Function: my_page().....	22
3.1.3 Function: submit_form().....	22
3.1.4 Function: main().....	23
3.2 Ransom_calc.py.....	23
3.2.1 Function: ransom_calc(params).....	23
3.2.2 Function: run_model(params).....	23
3.3 Utilities.py.....	23
3.3.1 Function: output_parser(params).....	23
3.3.2 Function: theta(params).....	23
3.3.3 Function: csv_parser(params).....	24
3.3.4 Function: to_list(params).....	24
3.3.5 Function: sample_gen(params).....	24
3.3.6 Function: hypothetical_fixed(params).....	24
3.4 Population.py.....	24
3.4.1 Class: population(params).....	24
3.4.2 Function: fitfunc(x, a, b, c).....	25
3.4.3 Function: q_finder(pt, a, lmbd).....	25
3.4.4 Function: Q_calc(v, WTP, n).....	25
3.4.5 Function: find_F(z,*params).....	25
3.4.6 Function: var_finder(v, WTP, n).....	25
3.5 Mal_model.py.....	26
3.5.1 Class: mal(params).....	26
3.6 Econ_model.py.....	26
3.6.1 Function: fixed_model(pop, m, a, lmbd, n, i).....	26
3.6.2 Function: fixed_opt_model(pop, m, a, lmbd, n, i).....	26
3.6.3 Function: fixed (this_mal,sample_WTP,i).....	27
3.6.4 Function: discrim (this_mal,sample_WTP).....	27

3.7	Summary.....	27
4	Optimal Model System: General Report.....	28
4.1	Scope of the Optimal Economic Model System.....	28
4.1.1	Objectives.....	28
4.1.2	Key Features.....	28
4.1.3	Measurable Impact.....	29
4.2	Mathematical Basis of the Optimal Model System.....	29
4.2.1	Deriving Population Variables.....	29
4.2.2	Deriving the Optimal Profit and Ransom.....	30
4.2.3	Deriving Threat Values.....	31
4.3	Summary.....	31
5	Conclusions.....	32
5.1.1	Future work.....	32
	References.....	33

List of figures

Figure 1 – Default View.....	12
Figure 2 – User Defined Population Mode.....	13
Figure 3 – Target Selection.....	13
Figure 4 – Ransom Type Selection.....	14
Figure 5 – Discrimination Type Selection.....	14
Figure 6 – Expanded Ransom, Price Brackets, and Costs Inputs.....	15
Figure 7 – Example Ransomware Selection.....	15
Figure 8 – Submit and Results Buttons.....	16
Figure 9 – Default Results Modal.....	16
Figure 10 – Simple Results Modal (Default pop, All, Fixed, \$100 ransom, \$10,000 costs, no examples).....	17
Figure 11 – Simple example compared with Wannacry (Default pop, Businesses, Fixed, \$600, \$10,000 costs)	19
Figure 12 – Complex Results with User Defined Brackets-based scheme compared with Hypothetical Cryptowall Example with Target Differentiation Capabilities (Individuals and Businesses).....	20

1 Introduction

This document presents three reports concerning the Optimal Economic Model System for Ransomware (OEMSR) version 1.0. This system is a locally hosted software client with a web-browser user interface. The OEMSR provides a simple, high-feedback means of modelling ransomware strains from the point of view of economic theory, providing a means by which scholars and forensic investigators can compare the effects of differing ransom strategies on specific population samples, to identify potential developments in the target-selection and price-selection phases of ransomware development. It also provides an abstraction of the threat-level posed by a given malware, determining not only how close to the optimal profit a given strain may be, but also showing how much of a risk the ransomware is to the whole infected population (regardless of whether they pay or are able to). The OEMSR is wholly focused on the economic factors at play in a ransomware attack but is intended to be extended to include ransomware propagation, trust systems and other relevant technological and psychological models in the future.

The first of the documents provided in this report is the user manual. This document provides instructions on the correct usage of the OEMSR. It also provides relevant examples and explains how users can define their own populations and (for more advanced users) malware strains. This user manual also includes several video entries provided with the OEMSR software as an audio-visual tutorial.

The second document is a functional definition of the OEMSR. This is a module-by-module, function-by-function breakdown of the back-end python code that forms the core of the OEMSR. This includes an overview of the Flask interface, which provides the connection between the back-end code and front-end HTML/JS implementation.

The third and final document details the mathematical model behind the OEMSR. The algorithms and equations at the core of population definition, optimal ransom calculation, profit derivation, and threat-level identification are all detailed in this section.

These three documents, collected into this one report, provide a complete overview of the OEMSR system. For both end-users and those interested in expanding the functionality of the system, this report provides the information required to do so within the parameters of the model system. Future work is discussed as a part of the concluding section of this report, focusing on potential modules that can be added to the system and relevance to other fields of inquiry related to the RAMSES project.

The structure of this report is as follows: Section 2 provides a user manual that details how the OEMSR can be operated. Section 3 provides a functional breakdown of the OEMSR by module and function, exploring the Python 3 back-end systems and some elements of the front end. Section 4 details the mathematical model at the core of the OEMSR. Section 5 concludes this report, summarising the three component documents and identifying future work. A copy of this document will be available on the RAMSES website [1].

2 User Manual

This document provides a user manual, which will instruct the reader in the installation, running and use of the OEMSR version 1.0.

2.1 Introduction to the OEMSR

The Optimal Economic Model System for Ransomware (OEMSR) is the final deliverable (4.4) of workplan 4 for the RAMSES project [3]. This system is intended to demonstrate the effects of target selection, ransom pricing and ransom strategies on the profitability and threat-level of a known or hypothetical ransomware strain. This will allow scholars and LEAs to analyse the optimal scenario and identify practices that may help disrupt the optimal path towards more effective and/or damaging strains prior to their emergence.

At time of submission, this system is suitable for identifying whether an intelligence network is required to achieve the optimal profit of a given malware, and whether the malware in question causes significant collateral damage (a large body of infections that the criminals have little intention of ever dealing with). Malware, in this context, refers to ransomware – a profit motivated form of cryptographic malware [2].

The scope of the OEMSR is Economic Modelling of Ransomware as a Business. It implements population analysis tools to turn survey data of willingness-to-pay values (the value individuals may associate with their files) into variables that allow the derivation of potential profit. This can be achieved for different ransom values and strategies. The aim of this process is to identify how close a selected strategy and price is to the optimal, and to characterise the traits that make a given strategy ‘optimal’. It also allows the comparison of strategies, to determine which combination of strategies and price-points will elicit the largest profit from a given population. Due to the definition of ‘optimal’ depending on the input population, it is not possible to define a blanket ‘optimal’ ransomware solution, but this tool allows for the identification of a ‘population optimal’ allowing for targeted solutions to be derived from OEMSR estimations, to protect at-risk populations based on real survey data. The model behind this tool is based on previous work by the University of Kent on economic theory and its applications to cybercrime [3][4].

This tool is a foundation for future development. Presently an Economic Model System, it is possible to extend this model with epidemiological models, trust networks and psychological models to explore other elements of the ransomware optimisation process.

2.1.1 Developers

The following individuals contributed to the development of the OEMSR version 1.0, listed with their roles and contributions:

- Darren Hurley-Smith
 - o Research Associate, University of Kent
 - o Deliverable management, Python 3 model implementation, Flask implementation, documentation, validation, and quality assurance
- Edward Cartwright
 - o Reader, University of Kent
 - o Economic theory, mathematical modelling, validation of the former, survey data
- Julio Hernandez-Castro
 - o Professor, University of Kent
 - o Cybersecurity context for applied economic theory, validation of the OEMSR
- Anna Stepanova
 - o Lecturer, University of Kent

- o Economic theory and survey data

2.2 Start-up, Dependencies and First-Run Guide

This section will guide you through the download, unpacking, dependencies installation and first-run processes of the OEMSR. All examples are for a Windows 10 machine and should also work on Windows 7.

2.2.1 Directory Set-Up

Follow these steps to set up OEMSR on your system:

1. Select an appropriate directory on your machine
2. `git clone https://github.com/DarrenHurleySmith/RAMSES_OEMSR.git`

This will provide a set of sub-directories and Python scripts.

Viewing the directory on your desktop, you will find 6 python files and 4 sub-directories. The first of these sub-directories ‘__pycache__’ can be ignored. ‘Static’ contains the css stylesheets, fonts, images, JavaScript and results sub-directories. No changes should be made to any of these directories, as the first 4 directories contain content vital for the front-end presentation and function. The results sub-directory contains image and text file outputs associated with any modelling performed using the OEMSR system, as a permanent record of one’s analyses.

Returning to the application top-most directory, ‘Templates’ contains a single ‘__layout’ html file. This may be changed to alter the appearance of the front-end but is not advised for inexperienced users.

‘Tutorials’ contains a series of video guides to using the tool, intended to supplement this user manual.

The 6 python files form the Flask presentation module and back-end logic of the OEMSR. These should not be altered unless modification by experienced Python 3 developers is intended. It is advised that you back up the OEMSR directory if you engage in development activities with the source code.

2.2.2 Installing Dependencies

As a Python 3 program, the OEMSR requires that you have Python 3.x installed. It also requires the following modules, installed using pip:

- Flask
- Wtforms
- Matplotlib
- Scipy
- Numpy

These can all be installed using the following instruction from commandline (assuming pip and Python 3 are installed):

```
> py -3 -m pip install flask wtforms matplotlib scipy numpy
```

Once installed, you are ready for your first run of the OEMSR.

2.2.3 First Run

To run the OEMSR, please ensure you have command prompt running at all times and use an appropriate browser. We recommend Firefox or Chrome, both of which have been tested with our front-end implementation.

To start the localhost:5000 server, type the following into command prompt and hit enter:

```
> py -3 model_front.py
```

This will elicit the response:

** Restarting with stat*

** Debugger is active!*

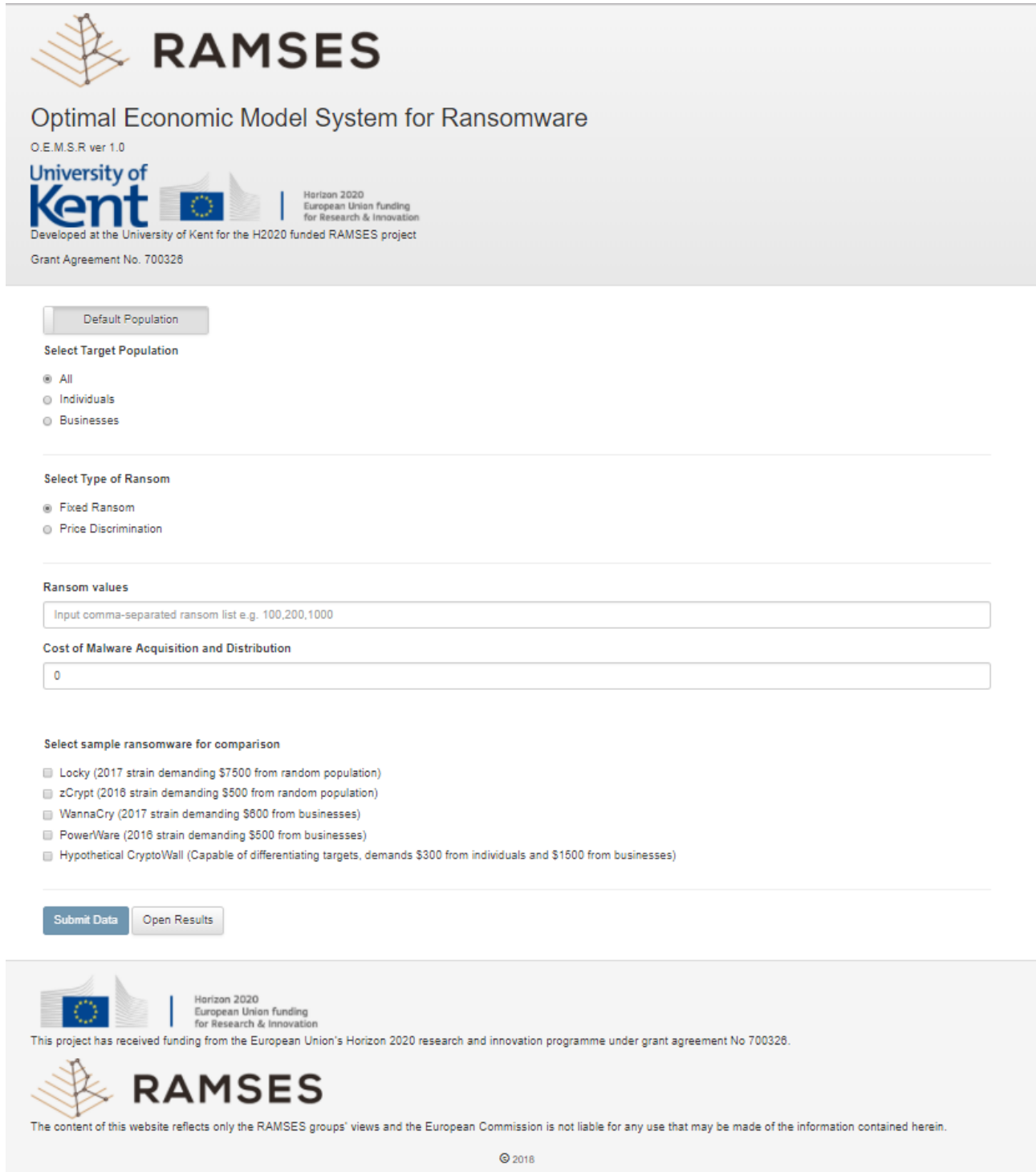
** Debugger PIN: 183-445-163*

** Running on <http://127.0.0.1:5000/> (Press CTRL+C to quit)*

This means that your OEMSR server is running. Go to your selected browser and navigate to localhost:5000. This will take you to the OEMSR homepage, the default selections page for ransomware and population definitions. You are now ready to define and analyse your first malware example.

2.3 Using the OEMSR

Figure 1 shows the default view that you will encounter when first starting your OEMSR client.



The screenshot displays the RAMSES web interface. At the top, the RAMSES logo is followed by the text "Optimal Economic Model System for Ransomware". Below this, it states "O.E.M.S.R ver 1.0" and "University of Kent". A European Union flag is also present, along with text indicating "Horizon 2020 European Union funding for Research & Innovation" and "Developed at the University of Kent for the H2020 funded RAMSES project". The Grant Agreement No. 700326 is listed.

The main section contains several input fields and buttons:

- Default Population:** A button labeled "Default Population".
- Select Target Population:** Radio buttons for "All" (selected), "Individuals", and "Businesses".
- Select Type of Ransom:** Radio buttons for "Fixed Ransom" (selected) and "Price Discrimination".
- Ransom values:** A text input field with the placeholder "Input comma-separated ransom list e.g. 100,200,1000".
- Cost of Malware Acquisition and Distribution:** A text input field with the value "0".
- Select sample ransomware for comparison:** A list of checkboxes for various ransomware strains:
 - ☐ Locky (2017 strain demanding \$7500 from random population)
 - ☐ zCrypt (2016 strain demanding \$500 from random population)
 - ☐ WannaCry (2017 strain demanding \$800 from businesses)
 - ☐ PowerWare (2016 strain demanding \$500 from businesses)
 - ☐ Hypothetical CryptoWall (Capable of differentiating targets, demands \$300 from individuals and \$1500 from businesses)
- Buttons:** "Submit Data" and "Open Results".

The footer includes the European Union flag, the text "This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 700326.", the RAMSES logo, and a disclaimer: "The content of this website reflects only the RAMSES groups' views and the European Commission is not liable for any use that may be made of the information contained herein." The copyright notice "© 2016" is also present.

Figure 1 – Default View

This is a simple web-page, with a small selection of smart features that allow the hiding of irrelevant fields for both input integrity and ease of viewing. There are several easily identified key elements to this, simplest, view.

2.3.1 Population Definition

The target population is the foundation of this system. Without a target population, the system would not be able to run at all, as it is the individual willingness-to-pay (WTP) of members of a population that allow the profitability of a given malware to be determined and the optimal to be found [3]. Willingness-to-pay is a variable derived by surveying the target population: in the case of our default population, a survey was conducted wherein individuals were asked questions relating to how much value they ascribed to their files. This provides an indication of how high a ransom they may be likely to pay to get those files back [4].

Figure 1 shows the population definition field in its default state, using the system-defined default population. This population is generated from a weighted list of WTP for individuals and businesses, with 1000 individuals and 100 businesses generated for a total population of 1100 for the model population. This is a suitable number as it allows for an acceptable degree of distribution of WTP values throughout the population, whilst not bogging down the model in lengthy generation and attribute derivation computations. Sub-types of individuals or business are not required: the willingness-to-pay of a given individual or business can be used as an indicator of their capacity to pay (which may in turn reflect personal or institutional wealth). Business default WTP values are always set higher than individual WTP, and the distribution of these WTP values can be found in the ‘ransom_calc.py’ module.

Figure 2 shows the alternative mode of this button.

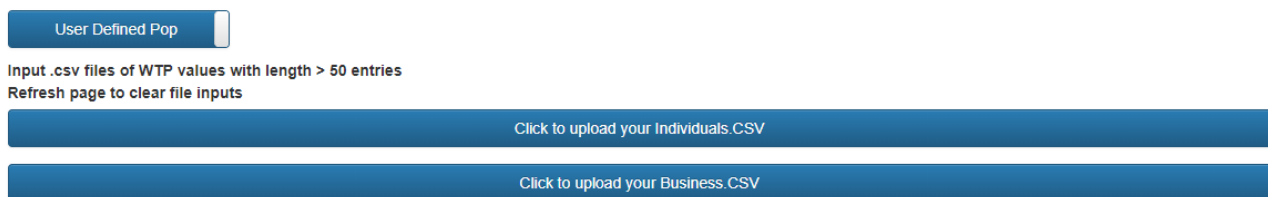


Figure 2 – User Defined Population Mode

User defined populations may be added from CSV files. Each population type (individual or business) should be input as its own, column format, csv file (each WTP on its own column). The only variables required in each CSV are the WTP values of the surveyed or hypothetical population. The resulting population will have a size equal to the number of WTP values: the population is purely defined by the CSV inputs in these cases. If you input one CSV, but leave the other blank, the default will be used for the uninitialized population type. To clear your file selections, refresh the page.

It must be stressed this this population is considered infected with the target malware unless the appropriate defence calculations say otherwise later in the OEMSR process. This is because the OEMSR doesn’t implement an epidemiological model at present and must assume that the population targeted has reached the stage at which money will, or will not, change hands. An epidemiological model is slated for future work.

2.3.2 Population Types

Having selected a target population, you must now determine the way in which your ransomware is intended to interact with it. You can select whether you will target businesses, individuals, or the entire population, as shown in Figure 3. The rationale behind these population types, is that some affected groups, such as businesses, may be significantly more likely to pay a high ransom [7].



Figure 3 – Target Selection

2.3.3 Type of Ransom

After selecting how your hypothetical malware will select the target population, you must select the type of ransom. Fixed, the default setting, is by far the most common type of ransom type in real malware. It is simple, requires no specific intelligence and aims to demand a set prices, sometimes with multipliers based on swiftness of payment (with increases for overdue payment). Also shown in Figure 4 is the price discrimination option. This is a more sophisticated type of ransom; wherein sub-strategies can be defined. These strategies detail how multiple ransom values are attributed to the population.

Select Type of Ransom

- ☐ Fixed Ransom
- ☒ Price Discrimination

Figure 4 – Ransom Type Selection

2.3.4 Type of Discrimination

Figure 5 shows the sub-types of discriminatory ransoms. This field will not show unless price discrimination has been selected previously. The options include non, population type, and price brackets.

Select Price Discrimination Type

- ☒ None
- ☐ Population Type
- ☐ Price Brackets

Figure 5 – Discrimination Type Selection

None will randomly assign different ransoms (defined by the user) throughout the target population. This represents poor or no intelligence gathering, and a scatter-shot approach to seeing if multiple ransom values can lead to higher profit. Up to four ransom values can be defined.

Discrimination by population type will allow the definition of two ransoms; the first being presented to the individual population, and the second to the business population.

Price brackets allows the definition of up to four ransoms and four price brackets. Brackets represent an estimation of WTP. For example, brackets (100,500,1000) will sort the population into WTP values of 0-100,101-500, and 501-1000. It is, therefore, not advised to have ransoms set higher than your bracket value for each bracket, as it is unlikely that members of that bracket will be willing to pay.

2.3.5 Ransom, Brackets and Cost inputs

Figure 6 shows the fully expanded field for ransom, bracket and cost inputs. Price brackets is hidden unless price brackets have been selected in the previous options.

Ransom values

Price Brackets

Cost of Malware Acquisition and Distribution

Figure 6 – Expanded Ransom, Price Brackets, and Costs Inputs

Ransom values are input and comma-separated integers or floats. For example, a fixed ransom may be 100, a discriminatory ransom (no strategy) might be 100,1000,2000, and a discriminatory ransom (pop type) might be 100,1000. If you put in too many values for the given settings, it will select values from the leftmost entry, until it has sufficient values. Attempts to use more than 4 ransoms will cause an overflow in the graphing portion of the model, do not attempt this at this time (InternalError:500 will be thrown).

Price brackets are entered in much the same way, you must have as many brackets as you have ransoms.

The cost of malware acquisition and distribution is a user defined floating point or integer value. It is a single value, equal to the estimated foxed costs of the malware. This is a purely user defined variable but can be expanded in future work to include elements from ransomware epidemiology, Ransomware as a Service (RaaS) data, and other additional modules that define the Optimal Model beyond Economic and Business considerations.

2.3.6 Example Malware Selection

Figure 7 shows the final selection the user may make before submitting their hypothetical ransomware to the OEMSR. Five malware examples are currently provided, with one hypothetical and four real examples. These malware strains have been selected from a variety of publications detailing the most widespread and publicly acknowledged ransomware for the last three years [5][6]. The user may select one or all of the options shown below:

Select sample ransomware for comparison

- ☐ Locky (2017 strain demanding \$7500 from random population)
- ☐ zCrypt (2016 strain demanding \$500 from random population)
- ☐ WannaCry (2017 strain demanding \$600 from businesses)
- ☐ PowerWare (2016 strain demanding \$500 from businesses)
- ☐ Hypothetical CryptoWall (Capable of differentiating targets, demands \$300 from individuals and \$1500 from businesses)

Figure 7 – Example Ransomware Selection

The first four are examples of contemporary malware, from within the last two years (at time of submission). These are based on real data and abstractions of their attributes. Fixed costs are estimated, as real data on the cost of development is extremely difficult to isolate. Further analysis of RaaS will be required to make accurate future estimations of price.

The last example is a hypothetical Crypto Wall implementation. We assume that the developers iterate on this crypto-ransomware, to include target profiling capabilities. We have included this option due to a lack of effective examples in the current literature: ransomware operators still largely send fixed ransoms, and when discriminating, discriminate by only targeting a certain portion of the population instead of setting tiered ransom prices based on assumptions regarding their infected population.

2.3.7 Results

Having completed all prior steps, the results are obtained by clicking Submit. In Figure 8, the submit button is greyed out because not all required fields have been filled. Filling these fields will allow the button to be pressed, which is indicated by the button returning to a more vibrant shade of blue.



Figure 8 – Submit and Results Buttons

Open results may be clicked regardless of whether the OEMSR has been run. However, the default display, shown in Figure 9, will be shown if there are no results to display. The last results generated will be shown, if the button is pressed after closing the modal view but before the next OEMSR process is run.

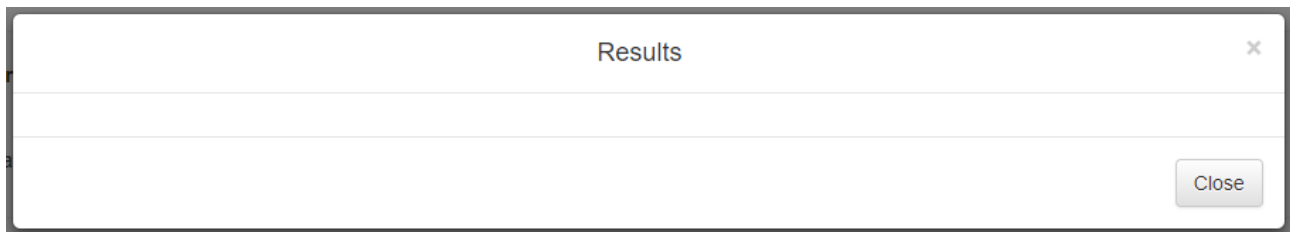


Figure 9 – Default Results Modal

Figure 10 shows a simple results output. This example runs just the user defined malware example over the entire default population. A fixed ransom of \$100 is demanded, with \$10,000 in fixed costs, and no examples are selected to compare against.

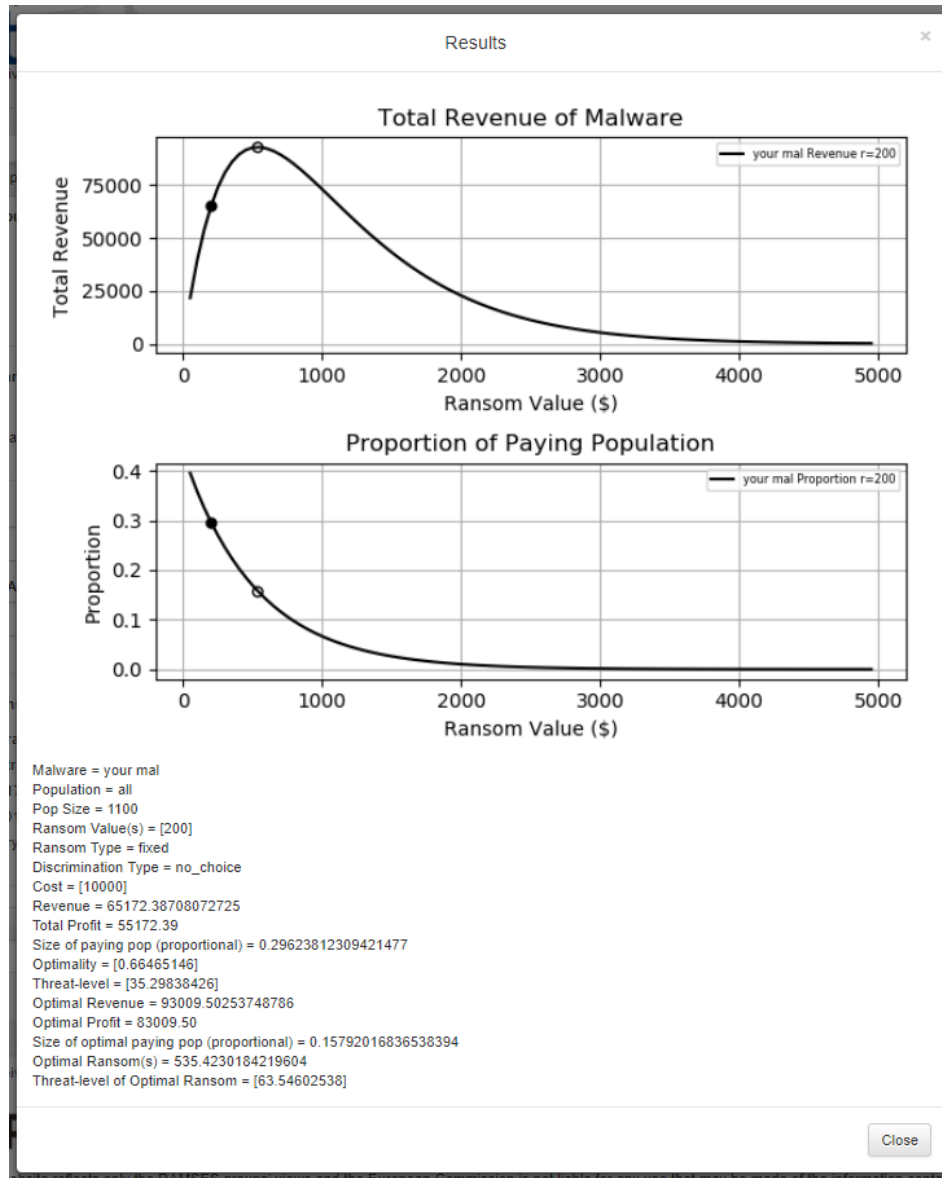


Figure 10 – Simple Results Modal (Default pop, All, Fixed, \$100 ransom, \$10,000 costs, no examples)

The graph outputs the continuity of ransoms from \$50 to \$4950, demonstrating the revenue generated and the proportion of the population that pays to generate that revenue. The filled marker represents the defined ransom results, the hollow marker represents the optimal. This is true of all lines; their respective filled and hollow markers will share the line colour.

A text output of the results, including variables not able to be depicted in the graphs, is provided. Importantly, this includes the profitability (revenue – costs), optimal profitability, and threat-level of the malware in question. Profitability is a good measure of attractiveness for a given malware; it represents the cash-in-hand outcome of the ransomware phase (before any costs associated with extraction of value to fiat currency via cryptocurrencies). The threat-level is a simple indicator of the amount of damage that a given ransomware might do, considering the proportion of people who pay, the population size and the total profit.

Threat is calculated as follows:

$$T = (i/n) * (1 - p_q)$$

Variable i represents the total profit, n is the total population size affected/targeted by the malware, and p_q represents the proportion of individuals who end up paying (thus contributing to i). This score is intended to represent the threat of the malware from a collateral damage and likelihood of repetition perspective. Malware that is attractively profitable, requires a small proportion of the population to pay, and therefore can

ignore a substantial portion of those they infect (leading to loss of files and disruption of business unless WTP rises to meet high ransom demands) is more of a threat than the alternative.

It should be noted that the optimal malware may be less threatening than a sub-optimal variant. This is because the optimality of a malware implementation is determined by its profit, not the damage it causes (which is a symptom of generating duress, not the intention). As a result, sub-optimal behaviours may cause malware to be more damaging, and the OEMSR allows users to investigate this further, by identifying where optimality is best served by being less damaging, and how the most threatening strains arise.

A general trend noted during testing was that optimal implementations generally had threat levels amongst the highest possible for their population and strategy combinations, but higher scores were possible if the decline in profit with sub-optimal paying proportions was sufficiently slow (as the larger population unable/unwilling to pay the ransom would begin to increase the threat faster than the profitability drops off).

Figure 11 (next page) shows the same user defined malware, compared against the WannaCry example. The two different population selection methods are easily seen, as the business focused WannaCry example exhibits a significantly different distribution of profitability and proportion of population willing to pay. The text output captures results from all the modelled malware, showing both the user defined and example malware.

Figure 12 (page after next) shows the output of a user defined price bracketed malware, and the hypothetical CryptoWall implementation that targets each population type with a different ransom. It must be noted that the line style changes with bracketed ransoms, to indicate which line is associated with each subdivision of the population (be it by bracket, randomly or by type). As previously shown, the text output is reproduced for all modelled malware strains. Where proportion, ransom and revenue differ by the population segment associated with a given form of discrimination, square brackets are used to enclose a list of the variables, to provide a breakdown of elements that go on to contribute towards the threat-level and profitability of a malware strain.

The calculation of the paying proportion is very sensitive to local minima in the population definition process of OEMSR. This results in a line that may exceed 1.0 times the population for some extremely dense WTP distributions, but it is not possible for the optimal ransom to ever exceed 1.0 for this value. Further study is required to isolate and smooth these local minima.

All results are saved to the `/static/results/graphs` and `/static/results/reports` directories for graphs and text reports respectively. All files are saved in the format `YYYY-MM-DD_UUID.png` (with reports taking extension `txt` instead). The UUID is tied to the report and graph, allowing outputs to be grouped long after the OEMSR session has been closed. These results are saved on every run of the OEMSR, so periodic cleaning of these directories may be required.

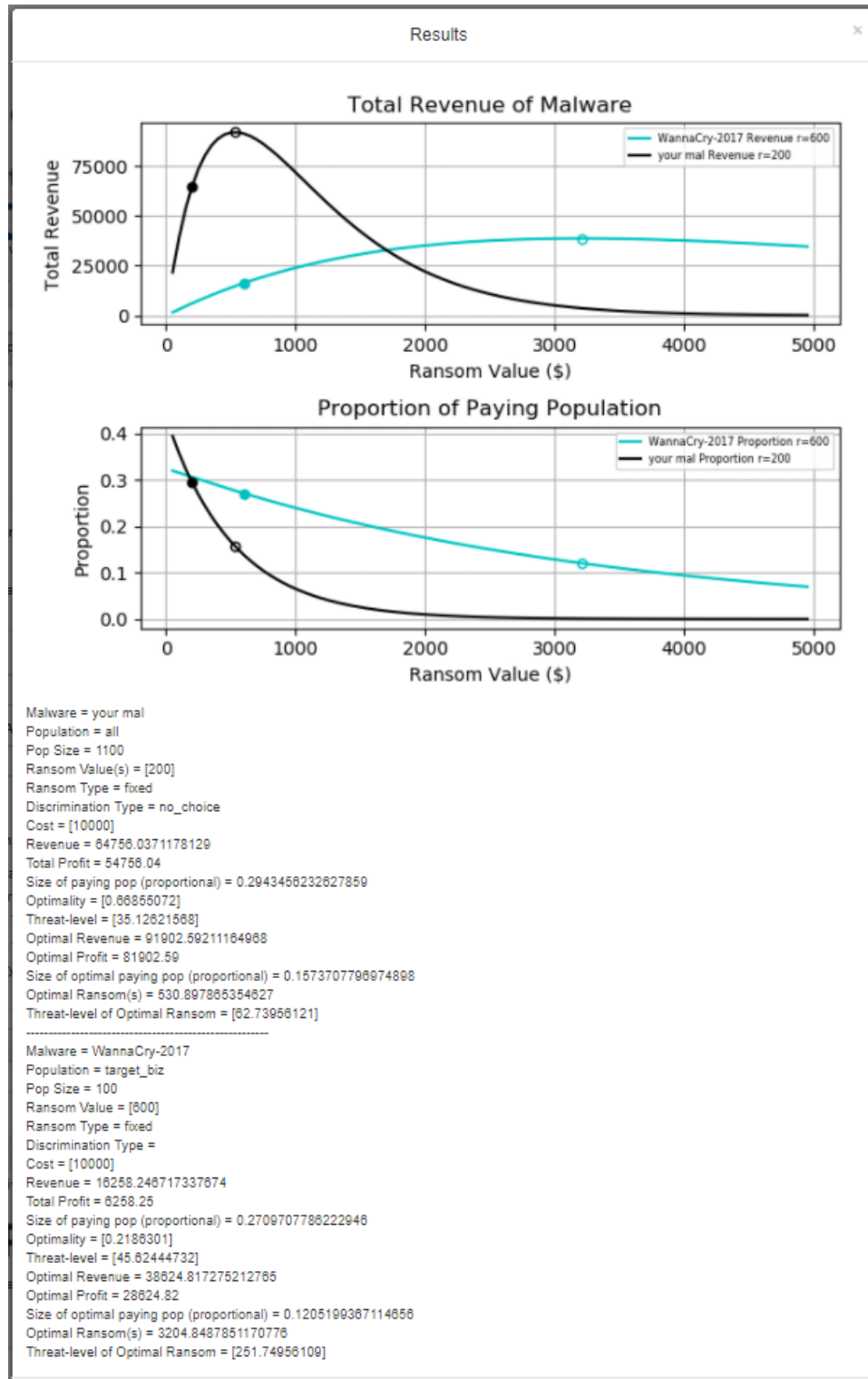


Figure 11 – Simple example compared with Wannacry (Default pop, Businesses, Fixed, \$600, \$10,000 costs)

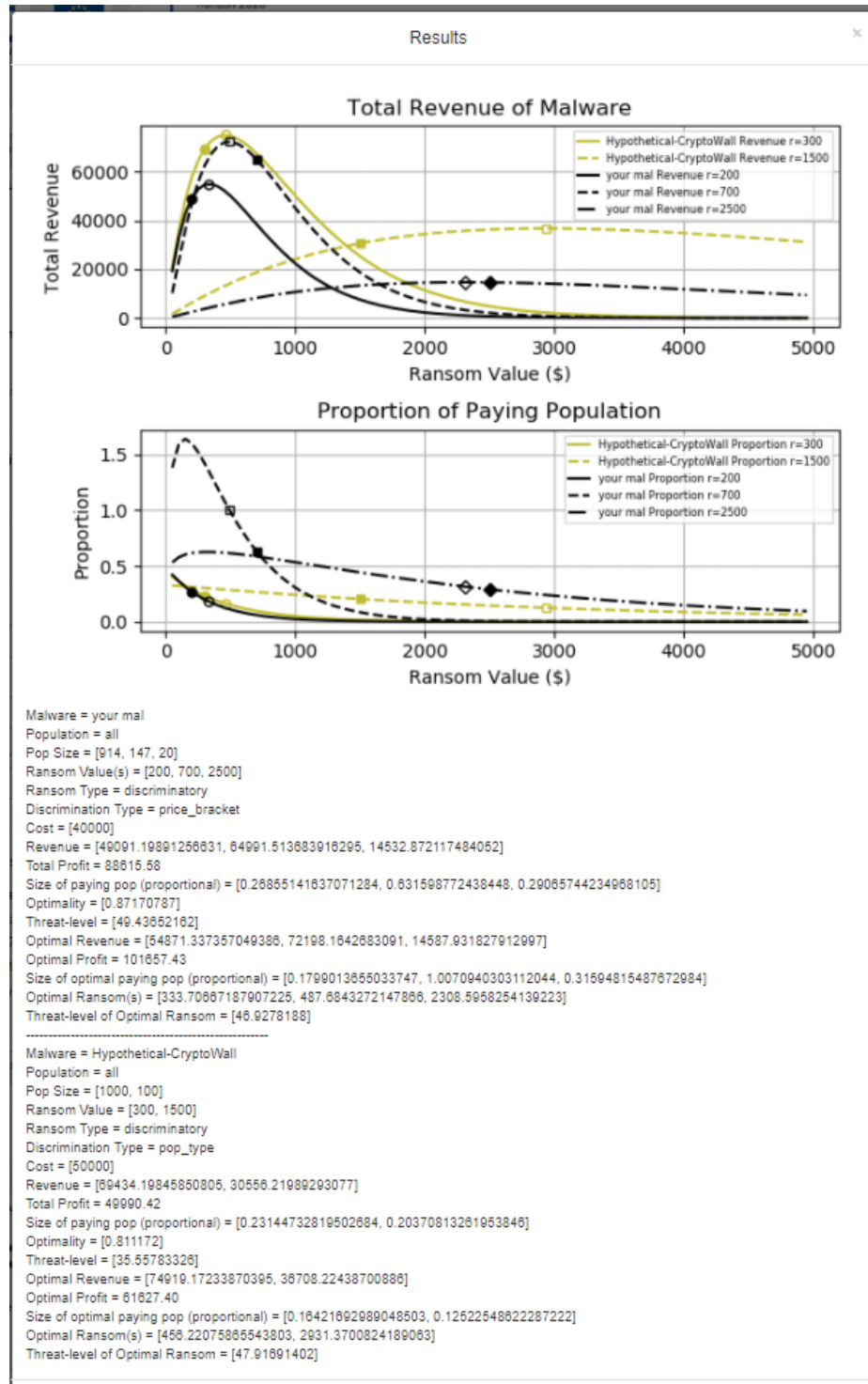


Figure 12 – Complex Results with User Defined Brackets-based scheme compared with Hypothetical Cryptowall Example with Target Differentiation Capabilities (Individuals and Businesses)

2.4 Summary

This section provides usage instructions, explanations of key processed and output definitions. The content has been aimed at users of low to moderate technical proficiency. More advanced users should refer to the functional definition and technical breakdown, which will provide more detailed information regarding the theoretical and programmatic aspects of the OEMSR.

3 Functional Definition of the Optimal Model System

This document provides a breakdown of the OEMSR by Python module, with each module being further sub-divided into its components functions. These are, in turn, described in terms of their dependencies and return values, to provide a full map of the software implementation of the OEMSR system. As the front-end may be replaced with PyGUI, WPF, or any other GUI framework, the focus of this documentation is the model-specific Python modules, with some reference to the flask implementation that forwards results to the front-end system.

All Python files are available in the OEMSR base directory for your reference and potential modification. Python 3 is the version selected for this work, it has not been tested under Python 2.7 and is not likely to be compatible.

3.1 Model_front.py

This is the module that parses the input from and output to, the front-end. Implemented in Python 3 and depending on flask and wtforms, model_front provides management services between the back-end model and the webGUI. Also calculates the threat-level of a given piece of malware (this functionality will be refactored to utilities.py later).

3.1.1 Class: Reusableform(Form)

Defines an instance of form. Forms are required to outline input and output objects for the front-end. This class generates a form with attribute 'validators=required' indicating that the form will check inputs to determine whether appropriate conditions have been met for parsing of input parameters. This provides a degree of error reporting.

3.1.2 Function: my_page()

Returns the render_template('_layout.html'). This is the default page for the OEMSR and primes the web GUI for further user interaction. Routes to the index url (/).

3.1.3 Function: submit_form()

Returns JSON converted variables; success, imageName and myResultsString. These variables are used to return image and text results to the results modular on the frontend. Frontend input is collected and processed into the following relevant variables:

- Population – list
- Ransom_type – string
- Discrim_method – string
- Ransom – np.float64 or list thereof
- Price_brackets – np.float64 or list thereof
- Fixed_costs – np.float64
- Handling_costs – np.float64 (not yet implemented, variable included for future reference)

These variables are taken as arguments by ransom_calc.ransom_calc(args). Ransom_calc returns a parameterised list of values which are used to populate the results text. A matplotlib.pyplot object is implicitly returned from within ransom_calc sub-functions, and is saved using a datetime and uuid generated unique file name (with .png extension).

This function also contains some error detection logic for out-of-scope user inputs that are not caught by the frontend logic. String handling for results output is also performed, though it is scheduled for refactoring into the utilities.py module for version 1.01.

3.1.4 Function: main()

Initialises and runs the flask application via app.run(). Takes no explicit arguments.

3.2 Ransom_calc.py

This module passes inputs from model_front to the run_model(args) function via ransom_calc(args). This module handles the selection of appropriate population types, or segments, providing appropriate initialisation variables for the economic model. It also initialises user define malware (this_mal) and example malware strains.

If the user has not defined a survey population, a default population is generated from a weighted list of ransoms (random assignment in a prescribed distribution).

3.2.1 Function: ransom_calc(params)

Checks whether user defined population is input, and of sufficient length. If not, generates default population of 1000 individuals and 100 businesses. Generates example malware, uses user definitions to generate this_mal. Invokes run_model(args), parses and returns input to model_front.py.

Params = usr_def_ind_WTP, usr_def_biz_WTP, target, price_model, diff_method, ransom, price_bands, fixed_costs, handling_costs, example

Returns = n, TR, qp, optTR, oqp, opt_r, e_n, e_TR, e_qp, example_mal_names, this_mal.n, e_r, e_c, e_tm, e_pm, e_dm, e_optTR, e_oqp, e_opt_r

3.2.2 Function: run_model(params)

Checks for this_mal target population and ransom type. Selects appropriate population list and runs econ.discrim(args) or econ.fixed(args) (where relevant) with suitable arguments. Also calls hypothetical_fixed(args) for discriminatory ransoms, a function that graphs the hypothetical distribution of ransom values over revenue and proportion of paying population. Returns the output of econ.discrim(args) and econ.fixed(args) to ransom_calc(args). Implicitly returns pyplot object from hypothetical_fixed(args), either directly or from nexted econ.fixed(args) call.

Params = this_mal, sample_ind_WTP, sample_biz_WTP

Returns = n, TR, qp, optTR, oqp, opt_r

3.3 Utilities.py

A utility module, designed to store functions that are used in multiple other modules. This is a toolbox module, in effect.

3.3.1 Function: output_parser(params)

Parses output files into human readable strings. These are passed back to model_front.py which then sends them through jsonify to allow output in the results model window. Also used to populate a report text file.

Params = n, TR, qp, name, r, c, pop, pm, dm, optTR, oqp, opt_r

Returns = output (a large string)

3.3.2 Function: theta(params)

Theta is a monotonically increasing function that expresses the phenomenon of diminishing additive returns. It is used to calculate the defensive and back up probabilities based on the WTP values of the target population. Defaults constrain this to a maximum d of 0.6 and maximum b of 0.3. For the default population this becomes d = 0.4 and b = 0.2. This will vary based on the relative WTP of the target population. Returns y, which represents either d or b depending on it's invocation (see econ_model_mod.py).

Params = e, m, c, a

Returns = y

3.3.3 Function: csv_parser(params)

Takes a csv file object as its argument and parses it to a list.

Params = x

Returns = y

3.3.4 Function: to_list(params)

Takes a form list object and parses it to a Python 3 list.

Params = x

Returns = y

3.3.5 Function: sample_gen(params)

Generates a sample population when no user defined csv is provided. WTP values (vals) are randomly assigned amongst a population of size n, with the distribution assigned by props. Returns a list.

Params = vals, n, props

Returns = sample

3.3.6 Function: hypothetical_fixed(params)

This function creates and populates a pyplot graph object. This will persist until saved and closed in model_front.py as a part of the form resolution process. When called, checks for ideal length of x-axis (ransom values) then plots for between 50 and ideal_max. Default max is 5000, but this will grow for larger ransoms.

Returns nothing, but the pyplot object is held implicitly until show() or close() are called on the object. This allows successive calls of hypothetical_fixed(args) to draw additional datasets on the same graph.

Params = pop, this_mal, a, lmbd, n, optTR, TR, qp, oqp, opt_r, j, r

Returns = none (implicit pyplot object created)

3.4 Population.py

Population generator module. Generates and modifies attributes relevant to the population, including the derivation of a, lmbd, d, and b, all of which are critical variables that define the receptiveness of a given population to certain ransom values because of the population WTP distribution. This module deals with the core mathematical functions of the OEMSR.

3.4.1 Class: population(params)

Defines a population by the proportion of backed up, defended machines, and willingness-to-pay (WTP).

Attributes: w, b, d

3.4.1.1 Function: __init__(self, wtp, backup, det)

Initialises an instance of population. Backup and det usually default to 0 until initialised by a call to defense_vars(args).

Returns = none

3.4.1.2 **WTP_parser(self)**

Parses the raw WTP list of the current population, invoking `var_finder(args)` to derive the population `a` and `lmbd` values. These values are critical to computing the optimal ransom and profitability associated with a given malware implementation. Takes no arguments, but operates on the variable `w` (a WTP list) associated with the class instance.

Returns = `a, lmbd, len(self.w)`

3.4.1.3 **Defense_vars(self, ad, md, cd, ab, mb, cb)**

When called, calculates the `d` and `b` variables associated with deterrence and backup effectiveness for the class instance. Uses parameters defined in the `econ_model_mod.py` module. Returns nothing, the `d` and `b` variables are modified locally and propagated to the local instance of the class. These variables correspond directly to the final proportion of paying members of the target population.

Variables `ad` and `ab` are the maximum values of `d` and `b` respectively (default 0.6 and 0.3). Variables `md` and `mb` are control values, defining the rate at which diminishing returns come into effect. Variables `cd` and `cb` define the minimum value of `d` and `b` respectively, defaulted to 0.

Returns = `none`

3.4.2 **Function: fitfunc(x, a, b, c)**

Returns = `a * np.exp(-b * x) + c`

3.4.3 **Function: q_finder(pt, a, lmbd)**

Takes the arguments `pt`, `a`, and `lmbd` and uses them to compute the proportion of the population who value their files more than `pt`. Returns this value as `q`.

Returns = `q`

3.4.4 **Function: Q_calc(v, WTP, n)**

Takes the arguments `v`, `WTP`, and `n`. Computes the value `Q`, representing the proportion of the population with a WTP at or higher than a given value. Returns `Q`.

Returns = `Q`

3.4.5 **Function: find_F(z,*params)**

Used as part of the grid search for `minimum(F)`, this function calls `Q_calc(args)` and `q_finder(args)`. Finds the value of `F`, which is returned as `F` to `var_finder(args)`.

`z = a, lmbd` (referenced as `x,y`)

`*params = v, WTP, n`

Returns = `F`

3.4.6 **Function: var_finder(v, WTP, n)**

Uses a brute force grid search with post-search optimisation function to find `minimum(F)`. This is critical, as for each population, the `a` and `lmbd` values associated with the `minimum(F)` value are required to accurately plot the proportion of the population willing to pay a given ransom. Returns `a` and `lmbd`, which are critical to all evaluation and optimisation functions of the model.

Returns = `a, lmbd`

3.5 Mal_model.py

A simple module, defining the class that defines malware instances in the model. Also provides two class-specific functions called under specific initial conditions.

3.5.1 Class: **mal(params)**

Defines an instance of malware. Attributes correspond to name, target_mode, price_mode, discrim_method, bands, ransom, fixed, handling, respectively. Called to create instances of user defined or example malware for use in the model system. Currently defines only the attributes relevant to the economic model, but may be extended trivially.

Attributes: n, tm, pm, dm, bd, r, c, hc

3.5.1.1 Function: **__init__(self, params)**

Initialises the attributes of the mal class.

Params = name, target_mode, price_mode, discrim_method, bands, ransom, fixed, handling

Returns = none

3.5.1.2 Function: **bands(self, temp_WTP, i)**

If price banding is selected, this function will be called to segment the population into the appropriate price bands. Is called in a loop to form new WTP lists, one for each band definition. Returns a new WTP list, this_WTP, which is representative of the newly banded population. Also returns temp_WTP, which has the banded population removed, to allow for iterative banding of the population.

Returns = this_WTP, temp_WTP

3.5.1.3 Function: **no_intel(self, temp_WTP, sample_WTP, i)**

Called to divide the population WTP list into equal length segments of random content. Due to the use of a PRNG, this content is usually possessing a distribution close to the normal value, resulting in near-identical segments, but this depends on the granularity and distribution of the target population. Used when price discriminating without a supplementary strategy. Returns a new list of lists, this_WTP.

Returns = this_WTP

3.6 Econ_model.py

A module containing only functions, this module provides the functions required to calculate the total revenue, profitability, and optimality of a given malware strain. Provides functions for both discriminatory and fixed ransoms, though there is potential for refactoring and cleaning of this module to condense the number of function calls throughout the program. Ties together mal_model and population, calling functions from both to provide estimations of the economic viability and optimality of the target malware(s).

3.6.1 Function: **fixed_model(pop, m, a, lmbd, n, i)**

Calculates the total revenue (TR) and paying proportion of the population (qp).

Returns = TR, qp

3.6.2 Function: **fixed_opt_model(pop, m, a, lmbd, n, i)**

Calculates the optimal total revenue (optTR), optimal ransom (opt_r), and optimal paying proportion of the population (oqp). Scheduled for future refactoring into fixed_model(args) due to redundancy. Originally implemented as a way to test that both sets of return values were calculated accurately and without cross-contamination.

Returns = optTR, oqp, opt_r

3.6.3 **Function: fixed (this_mal,sample_WTP,i)**

The top-most fixed-ransom function in this module. Called to calculate the current and optimal values for TR, qp and r. also returns a, lmbd, and pop to allow for comparative analysis of different populations, especially when modelling diverse population segmentation strategies.

Returns = n, TR, qp, optTR, oqp, opt_r, pop, a, lmbd

3.6.4 **Function: discrim (this_mal,sample_WTP)**

The top-most discriminatory-ransom function in this module. Called to calculate the current and optimal values for TR, qp and r. also returns a, lmbd, and pop to allow for comparative analysis of different populations, especially when modelling diverse population segmentation strategies. Used exclusively for discriminatory ransoms and calls appropriate population segmentation function from the population.py module as required by the malware specification. Calls fixed(args) multiple times, one for each population segment (corresponding to the number of ransoms and, if relevant, price brackets). Returns the same variables as fixed(args) but in list form to account for multiple executions of fixed(args).

Returns = t_n, t_TR, t_qp, t_optTR, t_oqp, t_opt_r

3.7 **Summary**

This section provides a functional definition of the OEMSR system. Each Python module has been broken down into its component classes and functions, with the arguments and return values made explicitly clear. Usage commentary has been provided where relevant, as well as suggestions for future improvements. This section will form a standalone document, which will be iterated on as the system is improved, revised and expanded.

4 Optimal Model System: General Report

This section of the report will define the theoretical work that went into the design of OEMSR's backend systems. It will also provide the design philosophy and rationale for choices made throughout the completion of this deliverable. The means of approximating population characteristics based on willingness-to-pay (WTP) values and the calculation of optimal profit using the derived variables are all explained in this section.

4.1 Scope of the Optimal Economic Model System

The purpose of this system is to model the economic aspects of ransomware, as identified and refined throughout deliverables 4.1, 4.2 and 4.3. This work represents the culmination of work plan 4 of the RAMSES project.

The OEMSR is currently in a version 1.0 state, ready for deployment as a local client application. The current OEMSR scope includes economic modelling of ransomware, assuming a population that conforms to user defined survey data or default parameters. The system is designed to take in such population data, along with the definition of appropriate historical malware strains, and determine the economic feasibility of the malware as a tool for profit.

The model incorporates several pricing strategies, including hypothetical population segmentation strategies. This allows for critical path analysis; which choices result in the most or least economically viable malware strains. It also allows researchers to identify whether this corresponds to the threat and damage potential of the malware in question. By identifying these attributes in advance of real world implementations, it may be possible to identify mitigation measures that reduce the efficacy of otherwise optimal ransomware implementations. This would allow LEAs to employ appropriate public outreach and technical campaigns ahead of any such optimal attempts to conduct cyber-extortion through ransomware. It is based on the work of Julio Hernandez-Castro, Edward Cartwright, and Anna Stepanova [3][4].

4.1.1 Objectives

The objectives of this deliverable were as follows:

- Develop a GUI application that allows users to economically model ransomware implementations
- Enable users to define a target population of arbitrary size and wealth distribution
- Enable users to define custom ransomware, in terms relevant to economic theory
 - Ransom value, type of ransom (fixed or discriminatory), discrimination strategy (where relevant) ...
- Provide visual feedback, text reports, graphs and saved data
- Plot not only the user's selections, but the optimal model for the population and ransomware characteristics chosen
- Give indications of the user define malware's proximity to the optimal, and the threat level of the proposed and optimal malware
- Allow comparison with historical examples of ransomware

These objectives have been completed and implemented in the OEMSR v1.0 system. There are several potential improvements and feature extensions that will be commented on in the future work section.

4.1.2 Key Features

The key features of the system are:

- Intuitive, familiar GUI presented via web browser (Chrome or Firefox recommended)
- Simple and clear presentation of results in the web GUI

- Results are saved in a local directory, including graphs and text reports of a given session
- 5 ransomware pre-sets are provided for comparative purposes, including 4 historically accurate and 1 hypothetical implementation
- A default population is provided for ease of use when lacking survey data to generate one's own population

4.1.3 Measurable Impact

The purpose of this tool is to provide a measurably impactful service to LEAs and scholars in appropriate fields. Below, we outline the measurable impact vectors, how they may be measured after submission of the OEMSR, and suggestions to increase impact further:

- Mapping hypothetical models to real world data from the perspective of scholars and LEAs would provide a means of scoring the current state-of-the-art in economic modelling of ransomware. This would, in turn, provide opportunities for further development of the model, either by having the current approach further reinforced by institutions outside of the UK, or by suggesting improvements to account for phenomena not yet encompassed in the general model.
- Allowing for general economic strategies to be derived for given ransomware strategies, LEAs and supporting institutions may more easily identify opportunities for targeted anti-ransomware campaigns. By identifying at-risk population sectors, appropriate strategies may be employed. OEMSR may aid in the development of countermeasures in advance of cyber-criminal transition to more-optimal modes of operation in the economic context. This could be measured through periodic surveys, especially in the wake of new attacks, to determine whether appropriate OEMSR-informed campaigns influenced WTP.
- By identifying population configurations, that the model deems 'hostile' (low profit), it may be possible to identify behaviours that are low-cost to individuals and businesses, but which have a high impact on WTP values, defensive values and back-up frequency (and habits). This could reduce the profitability of ransomware by depriving it of a viable population, which would be measurable through annual cyber-crime analyses.
- The impact of the model could be dramatically increased by further refining the frontend and eventually deploying it as a web-service, run by the University of Kent (or another appropriate partner).

4.2 Mathematical Basis of the Optimal Model System

The OEMSR version 1.0 is comprised of two key mathematical processes:

1. The derivation of the variables alpha and lambda (henceforth α and λ) from a list of WTP values associated with a surveyed or hypothetical population
2. The derivation of the optimal ransom, profitability and threat values, alongside the specific values for the user defined and/or example malware strains selected for modelling

A third consideration, the scoring of 'threat', is also entertained. Scoring threat is simple in form, but non-trivial in nature, as threat can be defined in many ways. Our implementation is defined in the appropriate sub-section. The equations that define the mathematical model at the core of OEMSR v1.0 are detailed in the three sub-sections that follow.

4.2.1 Deriving Population Variables

The population selected for use in the model is a critical element. It is vital for the calculation of revenue, profit, optimal versions of both and the optimal ransom. It also contributes critical variables to the calculation of the threat represented by a given ransomware. Although it is possible to assume the values of α and λ , this is unwise, as all subsequently derived variables will just be bound to the assumed population, instead of a defined one (be it real or hypothetical).

Starting with a list of willingness to pay variables, for a number (n) of individuals and/or businesses, we can let N denote the set of individuals. We can represent the WTP of individual $i \in N$ as v_i . The WTP of an individual directly corresponds with their personal valuation of files, important for calculating their likelihood of paying a given ransom. This forms a part of Selten's game theoretic approach to traditional ransoms [8], which has been adapted in this (and previous) WP 4 work to form the basis of our economic model. It should be added that we do not account for the effects of negotiation for the purposes of this system, as negotiation is shown to be a sub-optimal strategy by Cartwright et al (2017) in their extension of Lapan and Sandler's work on negotiation in hostage situations [9][10].

The equation below demonstrates the derivation of the proportion of individuals with WTP equal to or more than t .

$$Q(t) = \frac{|\{i \in N \mid v_i \leq x_t\}|}{n}$$

For a given value of a and λ , we can work out the fitted value for each valuation using the equation below.

$$q(t, a, \lambda) = p_t^a e^{-\lambda p_t}$$

An overall measure of goodness of fit is given by:

$$F(a, \lambda) = \sum_{t=1}^T (q(t, a, \lambda) - Q(t))^2$$

However, we need to find the values of a and λ that provide the minimum value F . This is achieved by performing a grid-search, in which the a and λ values are varied by increments between two boundaries. For a , the bounds are set at -0.5 and 0.5 with increments of 0.01. For λ the bounds are set at 0.0001 and 0.01, with increments of 0.0001. When the grid-search arrives at the minimum value of F , the a and λ values are an accurate estimate of the population's values of such, derived from their specific WTP distribution.

These variables may then be used for the calculation of the optimal model for a given ransomware strategy.

4.2.2 Deriving the Optimal Profit and Ransom

It is possible to determine how much people value their files, by fitting the demand function below:

$$q(p) = p^a e^{-\lambda p}$$

This is the proportion of the population that values its files more than p . If we set a ransom (r), then the total revenue may be expressed as:

$$TR(r) = r q(r) n (1 - b) (1 - d)$$

The variable n is the number of people attacks, b is the proportion with adequate deterrence expenditures, and d is the proportion with recent offline backups. These last two variables are currently expressed as a monotonically increasing, diminishing returns function of the WTP of the individual in question. This can and should be more accurately represented in the future, as its own willingness-to-spend value, as it is not always true that a higher WTP means more expenditure of defences. However, compared with our survey data, there is sufficient correlation to use this method as a means of computing a generalised value of b and d for the population in question.

The final calculation of the proportion of people willing-to-pay and needing to pay (because b and d indicate a proportion won't need to due to their defence spends) is computed at this point, as:

$$w = q(r) (1 - b) (1 - d)$$

The total profit for a specific value of r is given by the equation below:

$$\Pi(r) = TR(r) - F$$

F represents the fixed costs (such as acquisition and distribution of ransomware) incurred by the cyber-criminals. The optimal ransom (r^*), is given by:

$$r^* = \frac{1+a}{\lambda}$$

The optimal ransom can be used to compute the optimal profitability of a given strategy, as well as defining the proportion of individuals that will pay it. It is clear that a and λ are critical to this process, requiring their estimation through the rigorous process outlined in the previous sub-section.

4.2.3 Deriving Threat Values

For the OEMSR v1.0 system, threat is taken to represent the combination of attractiveness of a malware implementation (likelihood of copycat or repeat attacks due to high profitability for a given population) and low proportion of paying members of the population. The reason that a low-proportion is important, is that threat is an LEA/public perception issue, whilst profitability is all that matters to our hypothetical cyber-criminals. A low-proportion of paying individuals indicates that the ransomware in question doesn't need to honour most ransoms, simply because so few people are willing to pay. If it is optimal for this to persist, the successful, attractive ransomware attack will leave a swathe of collateral damage in its wake, in the form of lost files for those unable or unwilling to pay. Doctorow points out in his 2017 blog post that hackers selling DDoS attacks made significant money selling their services, indicating that there is a market for damage as well as honoured ransoms [11]. This suggests that here may be a motivation to reduce the total paying population that runs alongside profit-motive, though we retain our assumption that profit is the primary motivator for the attacks modelled by the OEMSR.

It is not true that all optimal ransomware is the most threatening, by this definition, but it is true that optimal ransomware, as defined above and in our previous deliverables, will have a high threat value due to its focus on minimising the paying population in favour of higher ransoms.

We define the threat-level (T) as:

$$T = \left(\frac{\Pi}{n} \right) (1-w)$$

This takes the profitability of the ransomware, divides it by the total population size, and multiplies it by the result of the proportion of paying individuals subtracted from one. This gives a score reflective of the assumptions described above. This score will increase with a lower proportion and higher profitability (relative the population size), but not to a degree where extremely profitable by high-paying-proportion ransomware doesn't generate an appreciable threat profile (which it should as part of threat is the attractiveness of the approach for repeat deployments).

4.3 Summary

This section of the report has provided an overview of objectives achieved, future objectives, measurable impact and the mathematical model at the core of the OEMSR v1.0 system. There is plenty of scope for future work, but the objectives of the RAMSES project's work plan 4 have been met by this deliverable and those that preceded it, in the manner outlined in this document.

5 Conclusions

An optimal model system, the OEMSR v1.0, has been developed in Python, and provides an interactive tool for LEAs and scholars alike. It provides a simple, intuitive interface, accessed through a web browser. User documentation, a functional definition and a general report of the objectives and mathematical basis of the model have been reported in this document, which represents a compilation of all relevant subject matter.

The user documentation provides a graphical and textual guide to the operation of the OEMSR tool. This is augmented with several video guides, each detailing a core element of the functionality of the system. These will provide support that will help users of varying proficiency to get the most out of the system.

The functional specification breaks down the backend code into its component elements. There is also some discussion of immediate improvements that may be made, after the submission of this deliverable. These improvements will be made iteratively, as this tool is likely to have a life cycle that exceeds the duration of the RAMSES project.

The general report details the objectives that have been observed and satisfied during the execution of this project. It also outlines the measurable impacts of the model system, alongside the other deliverables of work plan 4. The mathematical model at the core of the OEMSR is provided, with three specific elements discussed in detail: population attributes derivation, optimal and actual revenue, profit and proportion of paying population calculations (including defence expenditure abstraction), and the calculation of the threat posed by a given malware (and its optimal form).

In combination, these reports provide the information required to use, analyse and determine the relevance of system outputs to ongoing investigatory and scholarly work in the field of profit motivated ransomware. This model represents the current state-of-the-art in the economic modelling of ransomware as a business. It is our intention to take this tool further, by adding modules relevant to other projects dealing with elements such as malware epidemiology and psychological models of bargaining and duress, to enhance the tool over time and provide a living, iteratively improved model of optimal ransomware strategy, to allow for an increasingly rich depiction of how population and malware attributes lead to more viable ransomware implementations.

5.1.1 Future work

There are several elements of the OEMSR that would benefit from moving beyond the scope of the RAMSES project, but for which the resources and time could not be allocated.

- Epidemiological modelling of malware propagation would allow for better definition of the efficacy of defence expenses and backup strategies
- Operational costs should be implemented in an increment update, to represent the costs of dealing with paying customers (laundering money, support for those willing but unable to pay due to lack of technical knowledge)
- Further population type definitions may also be useful (such as government and military institutions)
- Incorporating a feedback-loop of information from public disclosure of ransomware wallet balances (such as @actualransom) would add value, by allowing us to validate our model during an attack [12]
- Building on previous work conducted by Professor Hernandez-Castro at the University of Kent [13], the model could be extended to model the optimal behaviours that criminals may express when faced with the very real possibility of being traced via the very cryptocurrencies they use to extort money from their victims. Meiklejohn et al demonstrate the efficacy of Bitcoin tracking to deanonymize users [14], whilst Sun et al. detail how anonymity-focused currencies (such as Monero) are defending themselves against perceived weaknesses in their current implementations [15]. This is a promising area of further innovation and application of the OEMSR.

References

- [1] <http://www.ramses2020.eu>
- [2] F-Secure. Crypto-ransomware. https://www.f-secure.com/en/web/labs_global/crypto-ransomware Last accessed: 29/11/2017
- [3] Hernandez-Castro, J., Cartwright, E. and Stepanova, A., 2017. Economic Analysis of Ransomware.
- [4] Cartwright, E. Hernandez-Castro, J., and Stepanova, A., 2017. Ransomware and Game Theoretic Insights on Kidnapping.
- [5] Talos Blog. *Ransomware: Past, Present and Future*. <http://blog.talosintelligence.com/2016/04/ransomware.html> Last accessed: 30/11/2017
- [6] David Bisson. Tripwire. Top 10 Ransomware Strains of 2016. <https://www.tripwire.com/state-of-security/security-data-protection/cyber-security/top-10-ransomware-strains-2016/> Last accessed: 29/11/2017
- [7] Kelly Kane. Phys.org. *Businesses more likely to pay ransomware than consumers*. <https://phys.org/news/2016-12-businesses-ransomware-consumers.html> Last accessed: 29/11/2017
- [8] Selten, R. (1988). *A simple game model of kidnapping*. In Models of strategic rationality (pp. 77-93). Springer Netherlands.
- [9] Lapan, H. E., & Sandler, T. (1988). *To bargain or not to bargain: That is the question*. *The American Economic Review*, 78(2), 16-21.
- [10] Lapan, H. E., & Sandler, T. (1993). *Terrorism and signalling*. *European Journal of Political Economy*, 9(3), 383-397.
- [11] Cory Doctorow. Two Hackers Selling DDoS. <https://boingboing.net/2016/11/28/two-hackers-are-selling-ddos-a.html> Last accessed: 29/11/2017
- [12] @actual_ransom. Twitter. https://twitter.com/actual_ransom?lang=en Last accessed: 30/11/2017
- [13] de Balthasar, T. and Hernandez-Castro, J., 2017, November. An Analysis of Bitcoin Laundry Services. In Nordic Conference on Secure IT Systems (pp. 297-312). Springer, Cham.
- [14] Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M. and Savage, S., 2013, October. A fistful of bitcoins: characterizing payments among men with no names. In Proceedings of the 2013 conference on Internet measurement conference (pp. 127-140). ACM.
- [15] Sun, S.F., Au, M.H., Liu, J.K. and Yuen, T.H., 2017, September. RingCT 2.0: A Compact Accumulator-Based (Linkable Ring Signature) Protocol for Blockchain Cryptocurrency Monero. In European Symposium on Research in Computer Security (pp. 456-474). Springer, Cham.

[end of document]